Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

ETSIINF

TRABAJO FIN DE GRADO

CloudRoom: Módulo de gestión de cursos online, abiertos y masivos.

Autor: Carlos Manuel García Suárez

Directora: Sonia Frutos Cid

MADRID, JUNIO DE 2014

Abstract

The reason of this Project is to develop the courses' module of CloudRoom, a Massive Online Open Courses platform. This module is encapsulated in a service-oriented architecture (SOA) based on a Cloud Computing infrastructure built on Amazon Web Services (AWS). Our goal is to design a robust Software as a Service (SaaS) with the qualities that are estimated in a product of this type: high availability, high performance, great user experience and great extensibility of the system. In order to address this, we carry out the integration of the latest technology trends in the development of distributed systems: Neo4j, Node.JS, RESTful Services and CoffeeScript. All of this, following a development strategy PLAN-DO-CHECK, using Scrum and practices of agile methodologies.

.

Resumen

La razón de este proyecto, es la de desarrollar el módulo de cursos de la plataforma de *Massive Online Open Courses* (MOOCs), CloudRoom. Dicho módulo está englobado en una arquitectura orientada a servicios (SOA) y en una infraestructura de *Cloud Computing* utilizando *Amazon Web Services* (AWS). Nuestro objetivo es el de diseñar un *Software as a Service* (SaaS) robusto con las cualidades que a un producto de este tipo se le estiman: alta disponibilidad, alto rendimiento, gran experiencia de usuario y gran extensibilidad del sistema. Para lograrlo, se llevará a cabo la integración de las últimas tendencias tecnológicas dentro del desarrollo de sistemas distribuidos como Neo4j, Node.JS, Servicios RESTful, CoffeeScript. Todo esto siguiendo un estrategia de desarrollo PLAN-DO-CHECK utilizando Scrum y prácticas de metodologías ágiles.

Índice

1 Introducción.

En los últimos 50 años nos hemos enfrentado a una revolución tecnológica que ha cambiado por completa la manera que teníamos de comunicarnos. Nos hemos convertido en depredadores de información. En la última década ya no basta con tener acceso a la información que circula por la red, sino que ahora, además, tiene que ser en cualquier momento y en cualquier parte. El fácil y rápido acceso a cantidades incontables de información ha dejado de estar al alcance solo de ámbitos académicos y corporativos, para formar parte de la vida cotidiana de un ciudadano del siglo XXI. Pues quien no ha abrazado todavía esta nueva forma de vida, parece ser de otro espacio y otro tiempo, parece ser el eco del siglo XX luchando por no ahogarse en el mar de la información, ya que como todo eco se ha convertido en nada más que ruido. Esta corriente ha dejado de ser eso, una simple corriente, para convertirse en un tsunami imparable que amenaza con no dejar ni el más remoto recuerdo de nuestros orígenes analógicos, y ahora ha centrado su atención en el templo más sagrado de transmisión de información, las aulas.

1.1 Crecimiento y auge del *Cloud Computing*.

Se podría decir que el *Cloud Computing* siempre ha estado ahí (Figura 1.1), pues en general puede ser tratado como un sinónimo para "red de computación distribuida". De hecho, el término "*the cloud*" es esencialmente una metáfora para Internet. Los vendedores han popularizado la frase "*in the cloud*" para referirse a software, plataformas e infraestructura que es vendida "*as a* service" (como un servicio) remotamente a través de Internet, es decir a través de una red de comunicación. Por tanto, como ya se mencionó más arriba, el concepto fundamental del *Cloud Computing* no es sino el de computación distribuida a través de una red de comunicación.

Este concepto de computación siempre ha estado ligado a la informática, pero lo que lo ha hecho evolucionar a lo que hoy en día llamamos *Cloud Computing* es el cambio en los consumidores de esta forma de computación.

Para conocer la evolución del *Cloud Computing* debemos remontarnos a sus orígenes, a lo que podríamos llamar la forma más "prehistórica" de este. Por consiguiente, empecemos situándonos en la década de 1950, cuando las computadoras centrales de

gran escala estaban reservadas a ámbitos académicos y corporativos. En este momento, para evitar tiempos de inactividad de estas costosas computadoras se empezó a permitir que múltiples usuarios compartiesen tanto el tiempo de CPU como el acceso desde múltiples terminales a las computadoras centrales. Como podéis ver el *Cloud Computing* ya estaba ahí, aunque restringido a grandes corporaciones que podían permitirse los centros de datos [1].

Y así ocurrió durante más de 30 años, hasta que en la década de 1990 las grandes compañías de telecomunicaciones empezaron a ofrecer servicios en *Virtual Private Networks* (VPN) con una calidad de servicio similar a la de los servicios que ya venían ofreciendo, pero a menor coste. Además, en esta década las computadoras empezaban a ser parte de la vida cotidiana. Esto hacía que científicos y tecnólogos explorasen nuevas formas de hacer disponible para más usuarios, el poder de computación a gran escala.

Desde el 2000, empresas proveedoras de servicios online como Amazon jugaron un importante papel en el desarrollo del *Cloud Computing*. Amazon buscaba modernizar sus centros de datos, ya que solo empleaba un 10% de la capacidad computacional de estos, y el resto lo usaba solo cuando había un pico de carga. Esta modernización llegó en forma de lo que hoy en día se entiende como una arquitectura *Cloud* para sus centros de datos. Esta nueva arquitectura trajo consigo una significativa mejora interna que les permitía implementar nuevas funcionalidades mucho más rápido.

Ya por el 2008, Gartner vio una oportunidad de negocio para el *Cloud Computing* "formar la relación entre aquellos que consumen servicios de IT, aquellos que usan servicios de IT y aquellos que venden servicios de IT", esto quedó plasmado en su famosa curva, por lo que ya nos encontrábamos ante los inicios de lo que hoy en día conocemos como *Cloud Computing* [2]. Una disponibilidad de redes de gran capacidad que se proveen como una utilidad con el simple acceso a Internet., y que ya por la década de 1960 se comparaba con el suministro de la industria eléctrica en el libro de Douglas Parkhill, *The Challenge of the Computer Utility* [3].

Hoy en día, la popularidad del *Cloud Computing* está en auge. La disponibilidad de redes de gran capacidad, dispositivos de bajo coste con capacidad de computación y almacenamiento y la adopción de virtualizaciones en el hardware, arquitecturas orientadas a servicios han dado lugar a la adquisición y crecimiento del *Cloud Computing.*

Los recursos que ofrece el hoy en día el *Cloud Computing* permiten desarrollar aplicaciones distribuidas capaces de abarcar un vasto número de usuarios a través de Internet, la red de redes, y en una gran variedad de plataformas y dispositivos a un bajo coste. Esto ha propiciado una gran caída en los tiempos de desarrollo de software

orientado a sistemas distribuidos y por consiguiente que la tendencia de este se haya movido hacia un desarrollo basado en la “nube”.

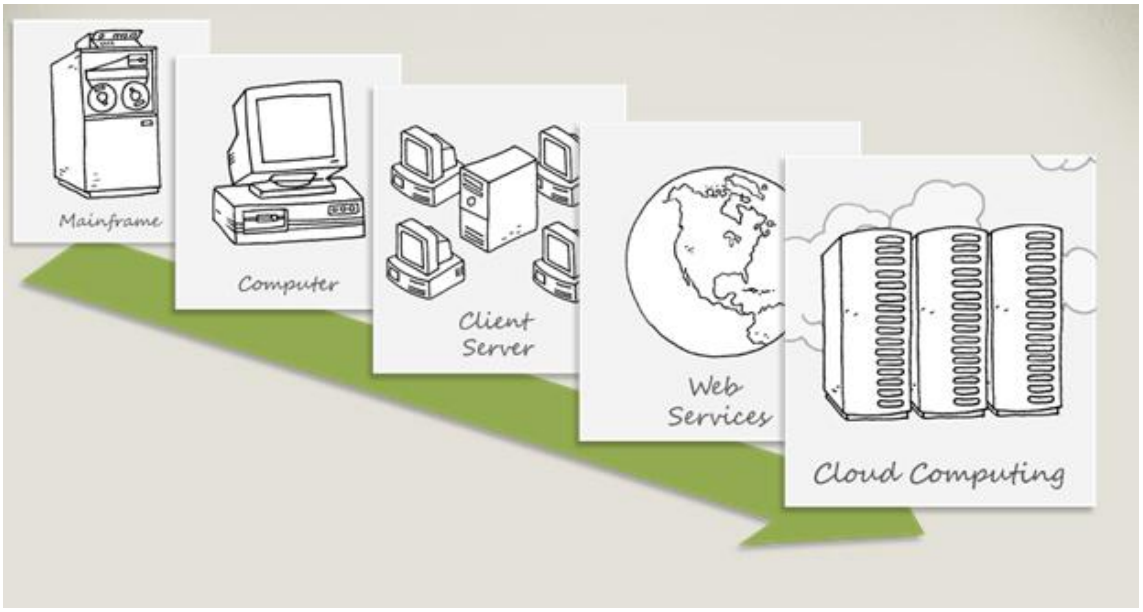


Figura 1.1 *Evolución del Cloud Computing*

1.2 Massive Open Online Courses (MOOCs).

Inmersos en medio de una corriente tecnológica como es el Cloud Computing, que facilita la escalabilidad de aplicaciones distribuidas con un rico contenido interactivo, era de esperar que tarde o temprano Internet y la tecnología mirasen hacia ese medio de interacción con información que engloba prácticamente a todos los usuarios de Internet, y no es otro que la educación.

Aunque la educación se conciba como un derecho fundamental para todos nosotros, durante la historia se ha topado con varios obstáculos que aún hoy en día podríamos encontrarnos, entre los que podemos destacar: prejuicios sociales, obstáculos geográficos u obstáculos económicos. Pero gracias a las nuevas tecnologías, el panorama para el futuro de la educación parece alejarse de lo que se concibe como educación tradicional.

A partir de ahora, las TICs serán fundamentales para el ahorro de gastos educativos, y permitirán que un mayor número de alumnos puedan matricularse a un menor coste y

sin importar el color de la piel, el sexo o la ubicación geográfica. Un cambio en el paradigma dela enseñanza que ya parece haber comenzado con la explosión en el año 2011 de las plataformas de MOOCs. Un producto de calidad y a un bajo precio. No es una estrategia novedosa. Es simplemente adaptar las nuevas tecnologías a un producto tradicional, y cuyo resultado es similar al que puede apreciarse en otros sectores. Además, esta nueva estructura permite el acceso a un mayor número de estudiantes de todo el planeta.

Un MOOC es un curso online con el propósito de tener una participación ilimitada y una vía de acceso a este que sea abierta a través de la web (Figura 1.2). A parte de los tradicionales recursos para el aprendizaje (videos, apuntes o conjuntos de problemas para resolver), los MOOCs proveen foros interactivos que ayudan a construir una comunidad para estudiantes y profesores [4]. Por último, cabe añadir que la estructura MOOC potencia ser autodidactas a grupos de personas que, de otra forma, tendrían muy difícil el acceso a estudios superiores.

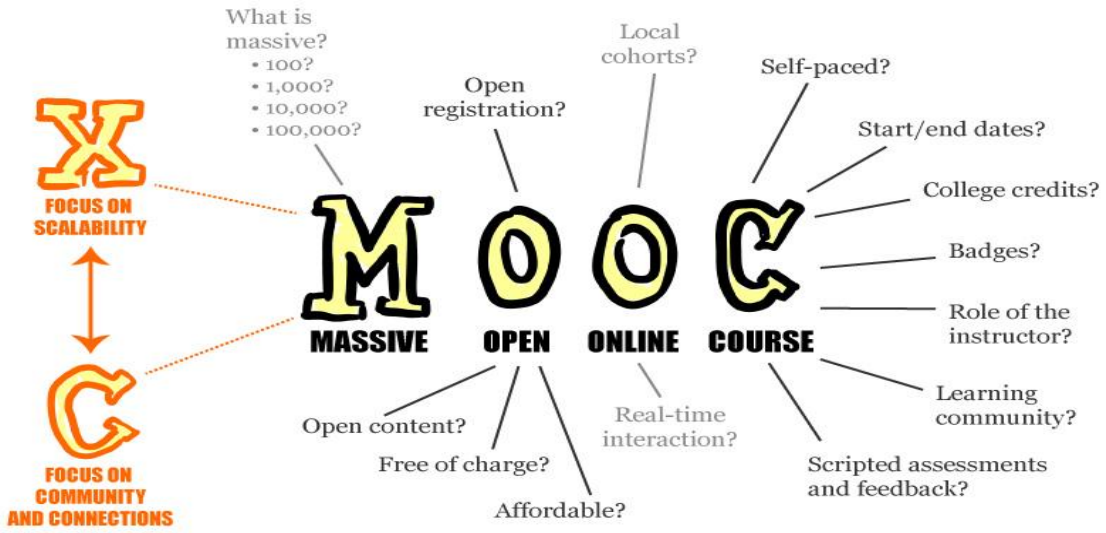


Figura 1.2 *Infografía de MOOCs*

Los Cursos Online Abiertos Masivos (MOOC) combinan lo mejor de las universidades (instrucción excepcional) con lo mejor de la tecnología (enseñanza interactiva).

1.3 Objetivos.

El trabajo consistirá en la realización del módulo de gestión de cursos de la plataforma CloudRoom. Una plataforma orientada a dar soporte a MOOCs (*Massive Open Online Courses*) que permite a alumnos, profesores e instituciones conformar una red de enseñanza con un fuerte componente social.

Para la realización del módulo de gestión de cursos será necesario, previamente, analizar el estado del arte tecnológico en el que los MOOCs se encuentran, así como estudiar las tecnologías con las cuales se pretende implementar dicho módulo. Estas tecnologías combinan el servidor Node.js basado en JavaScript y en una arquitectura orientada a eventos, la base de datos NoSQL de grafos Neo4j con su propio lenguaje de consultas Cypher y el servicio de almacenamiento cloud S3 (Simple Storage Service), conformando todas ellas el back-end de la aplicación.

Este proyecto estará orientado, por tanto, a desarrolla todas aquellas funcionalidades relacionadas con la gestión de contenidos didácticos y sociales, temarios, clases en vídeo, generación y evaluación de tests, foros, etc.

ESTADO DEL ARTE

"Hay que tener cuidado al elegir a los enemigos porque uno termina pareciéndose a ellos"

Jorge Luis Borges

2 Estado del Arte.

En los últimos dos años, los MOOCs (*Massive Open Online Courses)* han nacido, han ido adquiriendo una gran popularidad y el auge de estas plataformas ha supuesto un nuevo segmento de mercado en cuanto a SaaS se refiere. El aprendizaje gratuito y de calidad, ya que los cursos son impartidos por grandes instituciones de reconocido prestigio, fue lo que en un principio atrajo a los consumidores hacia este tipo de plataformas. Con el paso del tiempo, la calidad de este tipo de cursos ha ido creciendo de la mano del gran número de posibilidades que brinda una plataforma soportada en un entorno *cloud.* La flexibilidad, accesibilidad y disponibilidad que se busca en un MOOC ha pasado a primer plano y ha adquirido tanta importancia como el prestigio que las grandes instituciones de enseñanza proporcionan a los cursos ofertados.

Ya no basta con ofrecer cursos de las mejores universidades del mundo, ahora también cuenta, tanto o más, la calidad a la hora de impartirlo. Esto ha supuesto que las técnicas escogidas tanto para proveer de contenido a los alumnos como para recoger el *feedback* de estos se conviertan en un factor diferenciador con respecto de los competidores. Tanto es así, que las principales plataformas de MOOCs de la actualidad, apenas se diferencian en otra cosa que no sea la forma en que imparten los cursos, ya que el producto que ofertan, que son los cursos en si, suelen proveer el mismo contenido.

Por tanto, a continuación analizaremos de qué forma se estructuran y que técnicas siguen los módulos de cursos de cada una de las principales plataformas de MOOCs con el objetivo de conocer las principales funcionalidades de estos, para, más adelante, analizar las posibles debilidades y fortalezas de estos. De esta manera podremos orientar el desarrollo del módulo de cursos de CloudRoom para que nuestra plataforma, además de cubrir las necesidades del usuario objetivo, logre diferenciarse de la competencia que nos encontramos dentro del dominio de la aplicación, proporcionando así un software de calidad para el usuario final.

2.1 El panorama actual: E2E.

Hoy en día, los jóvenes tienen una probabilidad tres veces inferior de encontrar un empleo de la que tenían sus padres. Es decir, la tasa de desempleo de jóvenes se ha triplicado en los últimos 20 años, y esto se debe según un estudio de *McKinsey* titulado *Education to Employment: Designing a System that Works* [5] a la distancia existente

entre oferta educativa y laboral. Lo que significa que el problema no reside solo en la escasez de puestos de trabajo, sino también en que los jóvenes no encajan en el mercado laboral.

Los empleadores se están encontrando con puestos de trabajo vacantes debido a que no consiguen encontrar trabajadores con las habilidades necesarias. Y esto se debe a la falta de entendimiento del sistema educativo, las empresas, y los propios jóvenes por alcanzar metas comunes.

Pero obtener un consenso y dirigir la educación hacia la empleabilidad (*education-to-employment*, o E2E en inglés) supone varios problemas: el elevado coste existente en la educación superior (tasas, desplazamientos, alquileres…), los jóvenes creen que no salen bien preparados en cuanto a habilidades personales y la transición educación-trabajo se ve muy difícil.

Para paliar estos problemas los expertos confían en la innovación en educación apostando por el *life long learning* o aprendizaje continuo para guiar la educación en el futuro. La educación no consiste únicamente en grandes bloques teóricos y prácticos, sino que a veces consiste en pequeñas píldoras, capaces de combinarse con el resto de estudios, y que despiertan el interés de los alumnos.

2.2 MOOCs: Del Producto al Servicio.

Muchas cosas, y bien hechas, deben estar realizando los creadores MOOC, cuando solo en Coursera hay varios millones de personas inscritas en alguno de sus cursos. Millones de personas. Cientos de veces más de estudiantes que un campus universitario de tamaño medio. Esto quiere decir que los cursos tienen razón de ser, que existe una verdadera necesidad de conocimiento que va mucho más allá de los meros certificados, puesto que en muchos de estos cursos no se proveía de certificado alguno.

Llegados a este punto, me gustaría subrayar lo que en mi opinión ha sido el causante de la gran acogida de las plataformas de MOOCs durante estos años, y no es otra cosa que el producto que ofertan. Con producto me refiero al MOOC en sí. Los MOOC en plataformas como Coursera vienen con la firma de grandes instituciones de reconocido prestigio a nivel internacional como puede ser la Universidad de Stanford. Con un nombre así detrás de un MOOC, no es de extrañar que miles de usuarios en busca de conocimiento opten por ese curso.

Todos estos alentadores datos acerca de millones de personas en busca de conocimiento

se convierten en un problema cuando se presenta ante nosotros lo que está suponiendo el principal obstáculo de los MOOCs, este es la elevadísima tasa de abandonos de los estudiantes sin llegar a completar el curso, que alcanza el 96%. Según la Universidad de Rochester, que oferta sus cursos en Coursera, esto se debe más a la metodológica empleada para impartir un MOOC que a su propia esencia: la falta de conectividad entre alumnos, la imposibilidad de autenticar y hacer un seguimiento al estudiante… A todo ello podemos sumar la "calidad" de algunos MOOC, o incluso la inexistencia de tutorización. Todo esto es lo que podemos denominar como el servicio que proveen las plataformas de MOOCs, y es aquí, donde estas plataformas se han enquistado actualmente y donde invierten todos sus esfuerzos tratando de solucionar los problemas a los que se enfrentan.

La responsabilidad de las plataformas de MOOCs no es otra que proveer de una herramienta eficiente y eficaz a las instituciones que deseen ofrecer su producto, en este caso los MOOCs.

2.3 MOOCs: Plataformas actuales.

En las próximas páginas enumeraremos aquellas plataformas que más repercusión han tenido en el mundo de los MOOC.

2.3.1 Coursera.

Coursera se caracteriza por ser el MOOC con mayor número de usuarios del mundo. En torno a 4 millones de usuarios utilizan actualmente esta plataforma. Esto es así, debido en gran medida al gran número de cursos de universidades de prestigio que ofertan. Por tanto, esta plataforma se ha posicionado como líder actual, en cuanto a MOOC se refiere, y se ha convertido en la referencia donde fijarse para el resto de MOOC, los cuales desarrollan sus plataformas con un ojo pendiente de la filosofía con la que Coursera desarrolla nuevas funcionalidades para su gran comunidad de usuarios [6].

Esta plataforma se encuentra en desarrollo constante y busca continuar a la vanguardia en el sector. Para ello, ha implementado un gran número de funcionalidades que ayudan tanto a alumnos como a profesores a mejorar su experiencia de usuario, facilitando el aprendizaje y desarrollo en los cursos ofrecidos. Cuenta con varias funcionalidades diferenciadoras que pasaremos a explicar a continuación.

Presentación del curso.

Una vez te has registrado en el curso y este da comienzo, accedes al curso donde encuentras una página principal personalizada dependiendo del curso y la institución que lo imparte aunque con una estructura general para todos ellos (Figura 2.1). Esta estructura se basa en:

1. Un menú de navegación a través de todos los contenidos del curso situado en la izquierda a modo de barra lateral. En la barra de navegación aparece ordenado de arriba a abajo los menús que se deberían seguir facilitando al usuario la navegación de forma ordenada. Como se puede ver en la imagen encontramos una ayuda para empezar a navegar. En 'Start Here!' te hacen un resumen del curso, con la filosofía de aprendizaje que seguirá, el calendario con las fechas clave y la filosofía de calificación. Además aparecen todas las herramientas de las que dispondrás para seguir el contenido del curso y poder evaluar lo aprendido, las cuales se estudiarán en profundidad más adelante.

2. Densa descripción del curso acerca de los temas que tratará, las clases en video, los objetivos de aprendizaje, los ejercicios y laboratorios, o los foros de discusión.

3. Un panel lateral con las notificaciones del curso (fechas de entrega de trabajos, discusiones en el foro y próximos tests).

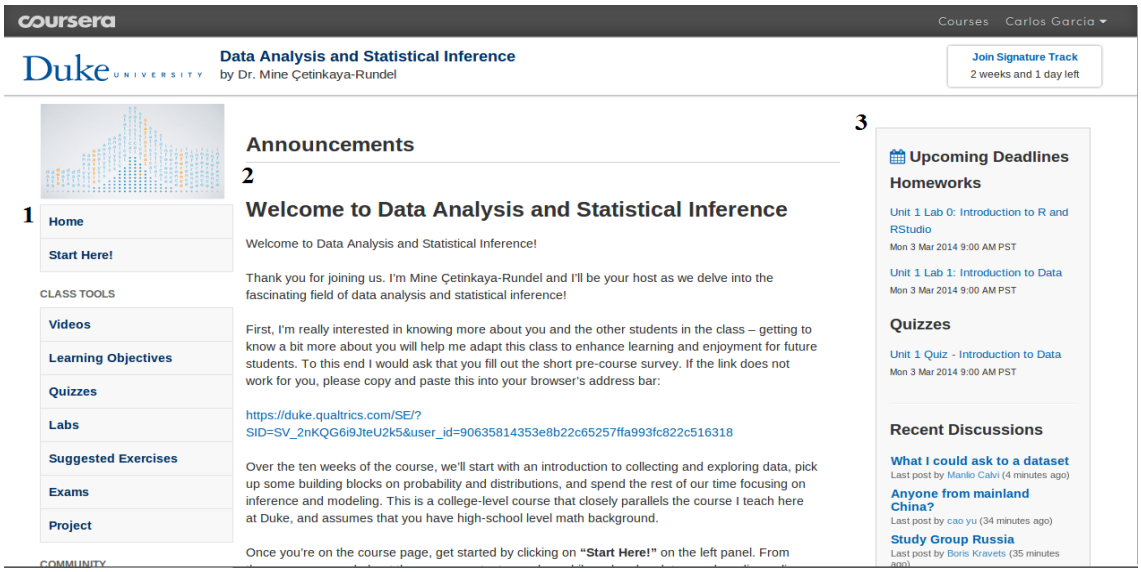


Figura 2.1 *Estructura presentación curso*.

Vídeos incrustados.

El aprendizaje mediante videos explicativos es la principal opción a la hora de seguir los cursos. Un menú con todos los videos te permite tener una visión de conjunto de todo el curso (Figura 2.2). Este menú te ofrece las siguientes posibilidades:

1. Descargar todo el contenido que aparece en el video en formato pdf (a modo de transparencias), los subtítulos en formato de texto o .srt, los tests que se realizan en el video en formato .pdf o descargar el video en formato mp4.

2. Permite al usuario reproducir el video que desee sin restricciones en cuanto al seguimiento del curso. Puede visualizar cualquier video sin tener que haber visualizado los previos.

3. Posibilidad de cambiar el reproductor de video utilizado en caso del que se está usando no funcione.

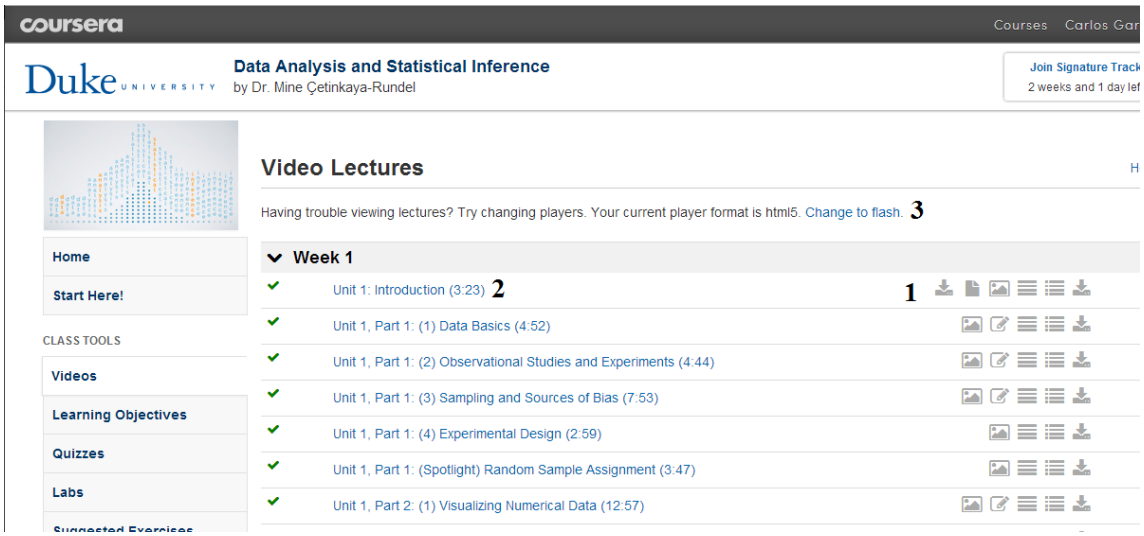


Figura 2.2 *Listado de los contenidos del curso.*

Por otro lado, la visualización del video proporciona funcionalidades muy útiles al usuario (Figura 2.3), las cuales son importante destacar:

1. Posibilidad de aumentar y disminuirla velocidad de reproducción del video (0.75x, 1x, 1.25x, 1.5x, 1.75x y 2x).

2. Aparecen marcados los momentos del video donde hay un test.

3. Puedes navegar al siguiente video o el anterior sin tener que salir del reproductor.

4. Hay tests con posibilidad de respuesta interactiva dentro del video, ofreciéndote la posibilidad de corregirlos en el momento o saltarlos.

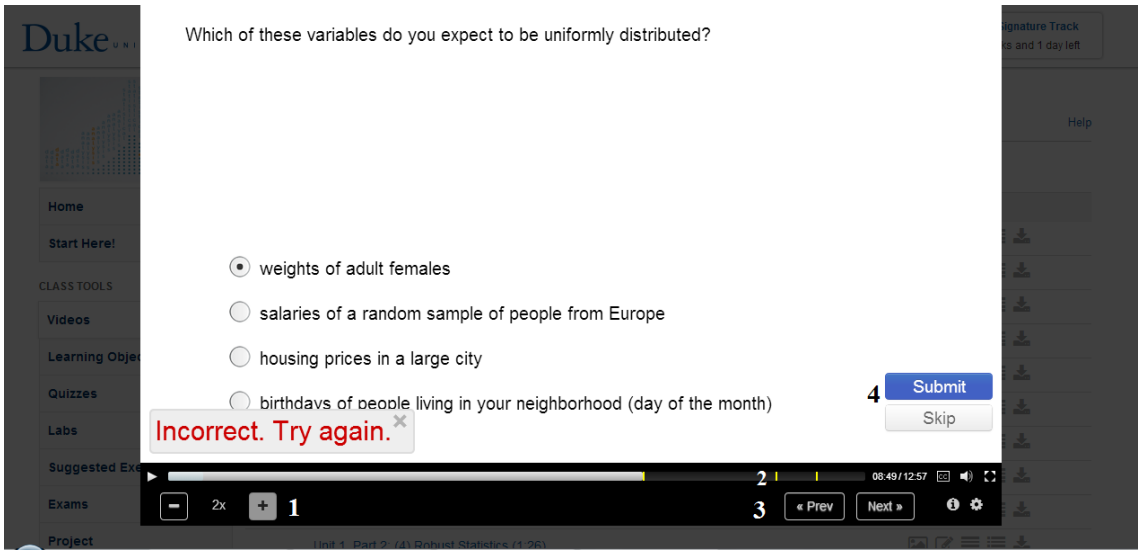


Figura 2.3 *Funcionalidades en el vídeo.*

Evaluación por compañeros.

Esta es una de las grandes funcionalidades escogidas por Coursera para llevar a cabo la corrección de los ejercicios y laboratorios del curso, cuya entrega y aprobación son imprescindibles para superar dicho curso.

Esta funcionalidad facilita mucho la corrección, ya que libera a profesores de corregir cientos de trabajos y a los desarrolladores de realizar herramientas de corrección para cada uno de los tipos de trabajo.

Además, proporciona un valor añadido al aprendizaje, pues te da la posibilidad de conocer las ideas que los compañeros tienen acerca de lo que has hecho. Esto resulta tan satisfactorio y útil como las correcciones de las propias instituciones.

Desde el punto de vista del estudiante, y por tanto como experiencia de usuario, se desprenden varias observaciones en cuanto a esta funcionalidad (Figura 2.4):

1. El flujo de tareas ayuda al alumno a la hora de realizar una evaluación justa de su trabajo, ya que este debe evaluar primero a sus compañeros y luego a si

mismo, lo que permite construir un criterio evaluativo antes de evaluarte a ti mismo.

2. El resultado obtenido es una combinación de ambos resultados (evaluación por parte de los compañeros y por tu parte), aunque en ningún momento se habla del peso de cada nota.

3. Explicación del tipo de trabajo a realizar.

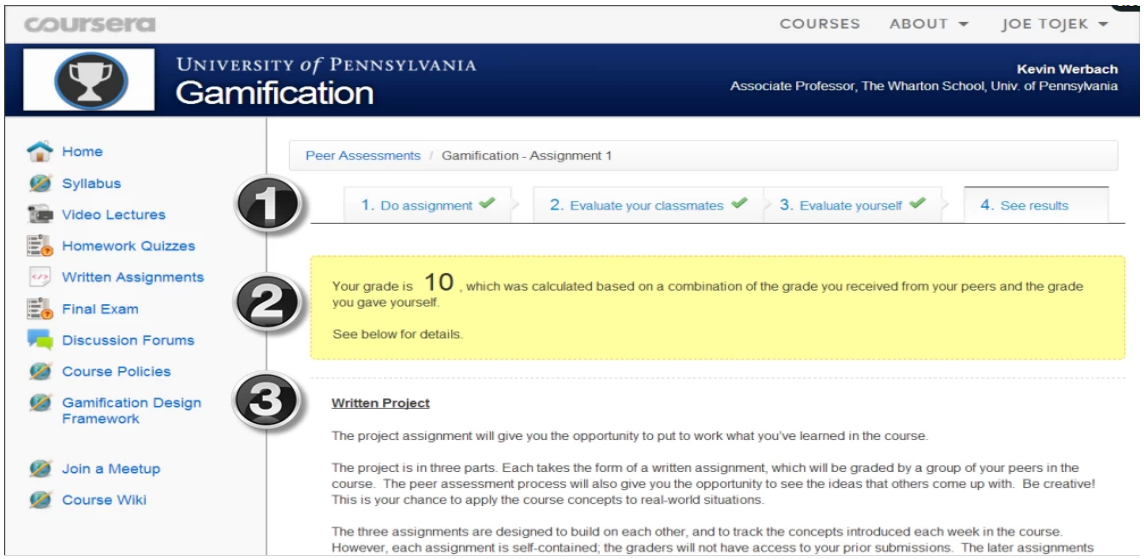


Figura 2.4 *Evaluación por pares.*

Especializaciones.

Esta ha sido la última gran funcionalidad añadida por Coursera. Proporciona la posibilidad de seguir varios cursos para finalmente obtener un certificado original por parte de la institución que te acredite para el campo estudiado.

Cabe mencionar que estas especializaciones son todas de pago, ya que te otorgan un certificado al final del curso. Dicho certificado posee las siguientes atribuciones:

- **Oficial**: Está aprobado por Coursera y la institución que lo imparte.
- **Verificable**: Coursera proporciona un una URI con un ID único que verifica dicho certificado.
- **Compartible**: Coursera proporciona una URL única que permite que cualquiera vea tu certificado.

Para verificar el seguimiento de los cursos y así garantizar la validación de los conocimientos, y evitar la pérdida de valor de las especializaciones debido a las trampas que pudieran cometer los usuarios en la realización de los cursos, Coursera ha introducido una *Signature Track.*

Signature Track.

La *Signature Track* es un sistema de verificación de identidad que usa una combinación de verificación de foto y patrón de escritura para confirmar tu identidad cada vez que entregas un trabajo que esté sujeto a evaluación en tus cursos (Figura 2.4).

Para crear esta firma se pide al usuario un ejemplo de tecleo para obtener el patrón de escritura de este, autorretrato tomado con la webcam y una fotografía de tu tarjeta de identificación oficial como puede ser el DNI (se garantiza el seguro almacenamiento y borrado de esta fotografía de la BD).

Este sistema de verificación se usa para cualquier curso que otorgue un certificado, ya sea una especialización o un curso solamente.

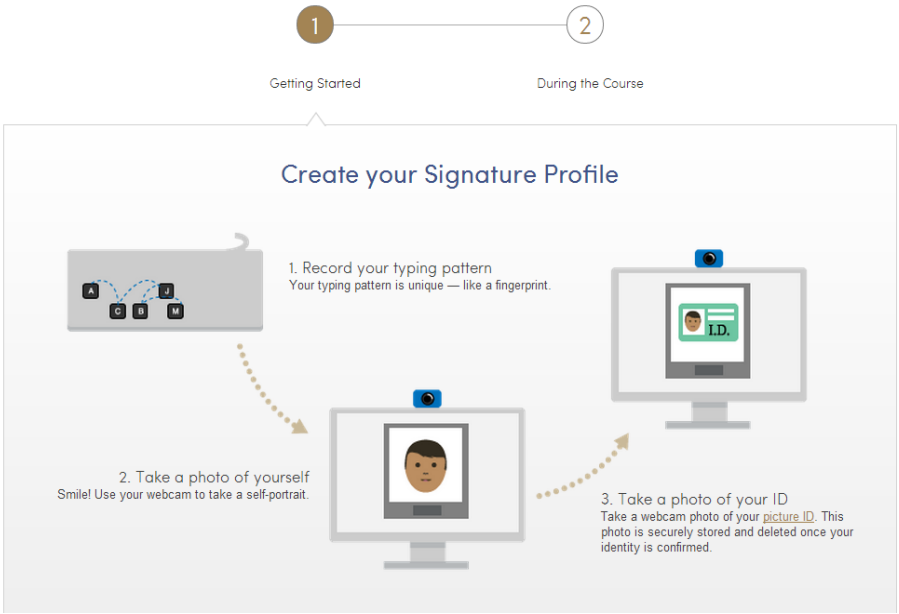


Figura 2.5 *Signature Track*

Trabajos en grupo.

En este aspecto Coursera todavía no ha introducido ninguna funcionalidad que mejore la experiencia de usuario, convirtiéndose en una de las debilidades de la gran plataforma, la cual tuvo que suspender un curso en Febrero de 2013 debido a los problemas técnicos que surgieron a la hora de crear grupos de trabajo para afrontar los ejercicios del curso.

Los principales defectos que se identificaron en el curso fueron:

- Defectos técnicos: La forma en la que te tenías que unir a un curso era a través de una hoja de cálculo de Google. Pero el servidor de Google no pudo soportar el gran volumen de tráfico.

- Falta de instrucciones: Los usuarios se quejaron de la vaguedad de instrucciones y la falta de claridad en cuanto a por qué había que unirse a un grupo o cuál era la utilidad de esta.

La no disponibilidad por parte de Coursera de funcionalidades orientadas al trabajo colaborativo en grupo o que intencionalmente se delegasen estas responsabilidades a los creadores del curso propició el cierre de este.

2.3.2 edX.

Se trata de una plataforma sin ánimo de lucro compuesta por instituciones educativas líderes a nivel mundial, el xConsortium. EdX [7] fue fundada por el Instituto Tecnológico de Massachusetts y la Universidad de Harvard y ya cuenta con más de 2 millones de usuarios. Desde su fundación en mayo de 2012, edX ha estado comprometida con una visión *open source* para expandir la educación online. EdX tiene su código liberado bajo la iniciativa Open edX [8] para que desarrolladores de todo el mundo trabajen en crear la siguiente generación de aprendizaje online. Esto dota a edX de una gran comunidad de desarrolladores que mejoran la plataforma de forma continua.

Principales características.

EdX cuenta con una interfaz muy depurada y a la hora de seguir un curso como alumno, en seguida te das cuenta de que es asombrosamente intuitivo y fácil de utilizar, y pasados unos minutos la interacción con la aplicación es muy fluida. La navegación está bien estructurada por lecciones en una barra lateral y una vez dentro del contenido de

estas puedes navegar por diferentes tipos de contenido mediante una barra horizontal que te muestra iconos del tipo de contenido que se mostrará (Figura 2.6). Y el *feedback* que se le muestra al usuario, en cuanto a donde se encuentra es muy útil y fácil de distinguir.

En cuanto a las funcionalidades, cuenta con todo el abanico de funcionalidades que se puede encontrar en otras plataformas de este tipo como Coursera, aunque salvando las distancias con las funcionalidades lucrativas de esta (certificados, etc.).

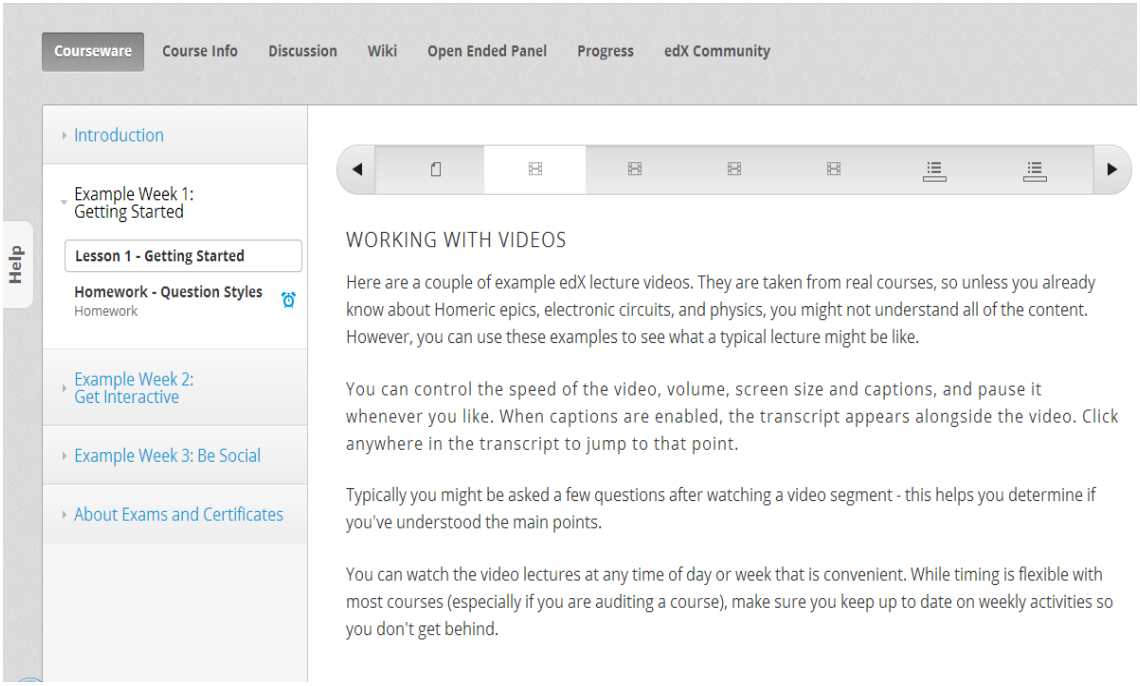


Figura 2.6 *Estructura curso edX*

Como profesor también te encuentras con un gran grado de sencillez a la hora de crear y estructurar un curso, convirtiendo así el tedioso trabajo de subir un curso a la plataforma virtual en algo al alcance de cualquiera [9] (Figura 2.7).

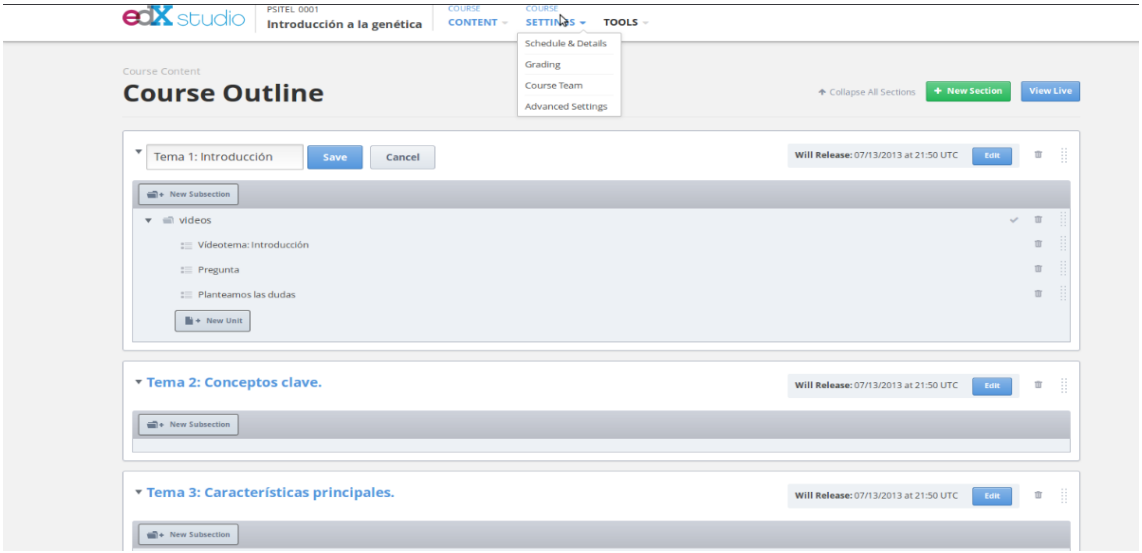


Figura 2.7 *Herramienta creación cursos***.**

Luego te puedes encontrar con otras opciones para configurar el curso que merecen la pena destacar como el apartado para definir los grados de superación del curso. Destaca por su flexibilidad y porque permite establecer los grados de forma visual desplazando simplemente la barra de cada grado hasta el objetivo deseado (Figura 2.8).

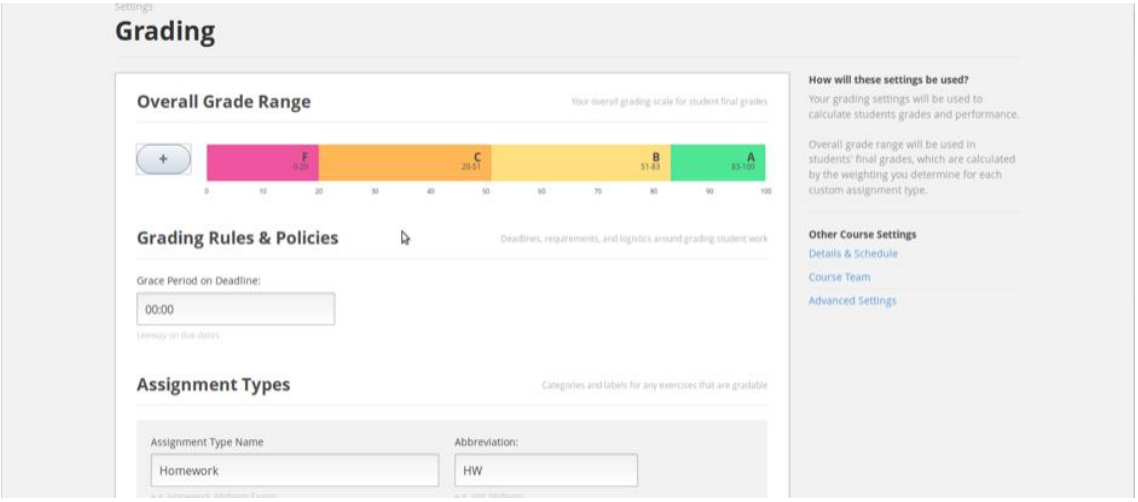
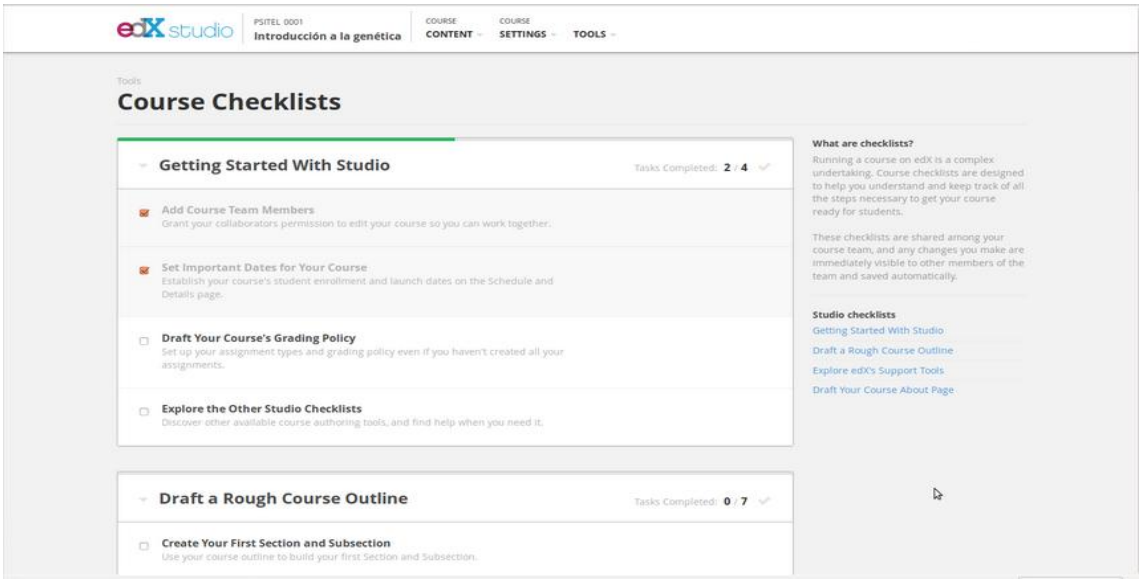


Figura 2.8 *Elección de pesos de calificaciones***.**

Otro apartado que conviene destacar es que el módulo de creación de cursos está dotado de una *checklist* de puntos a modo de guía metodológica que debes ir rellenando para lograr una creación completa del curso.



2.3.3 Udacity.

Udacity es una plataforma con ánimo de lucro que ofrece cursos online masivos y abiertos (MOOCs). Es el resultado de las clases de informática gratuitas ofrecidas en el año 2011 a través de la Universidad de Stanford. En la actualidad cuenta con 400.000 usuarios.

Los cursos en Udacity están orientados a la rama científica, con lo que podemos encontrar sobre todo cursos de matemáticas, física y sobre todo informática. Una de las fuertes bazas de Udacity son sus cursos especializados que cuenta con la colaboración de grandes empresas como Google, Nvidia, Microsoft, Autodesk, Cadence Design Systems, y Wolfram Research [10].

Principales características.

El formato del curso es más sencillo que el que nos podemos encontrar en plataformas como Coursera o edX. Cada curso consta de varias unidades que comprenden clases de vídeo con subtítulos *Closed Caption* (subtítulos ocultos), junto con cuestionarios

integrados para ayudar a los estudiantes a comprender conceptos y reforzar las ideas, así como tareas de seguimiento que promueven el modelo "aprender haciendo". Las clases de programación utilizan el lenguaje Python, las prácticas de programación son clasificadas por los programas de clasificación automatizada de los servidores de Udacity.

Hay que destacar la interfaz de usuario para alumnos, la cual no produce ningún cambio de contexto en la navegación y presenta sencillez a la hora de interactuar y a la hora de aprender a utilizarla. En pocos minutos el alumno es capaz de realizar todas las interacciones para realizar una tarea de forma fluida.

La interfaz cuenta con una barra de navegación lateral a la izquierda que muestra las distintas categorías de un curso mediante iconos (1)(Figura 2.9). A continuación, a su derecha, se encuentra el panel central con la plataforma de vídeos, un botón en la parte superior que despliega un índice de lecciones para navegar entre estas (2)(Figura 2.9), y una barra horizontal de navegación a través de los vídeos de la lección actual que además actúa como barra de progreso de esta, que muestra como *feedback* los vídeos que ya has visto, así como punto en el que te encuentras (3)(Figura 2.9). Finalmente, tenemos una columna a la izquierda que muestra las discusiones acerca del contenido que ves en ese momento y te permite iniciar una fácilmente (4)(Figura 2.9).

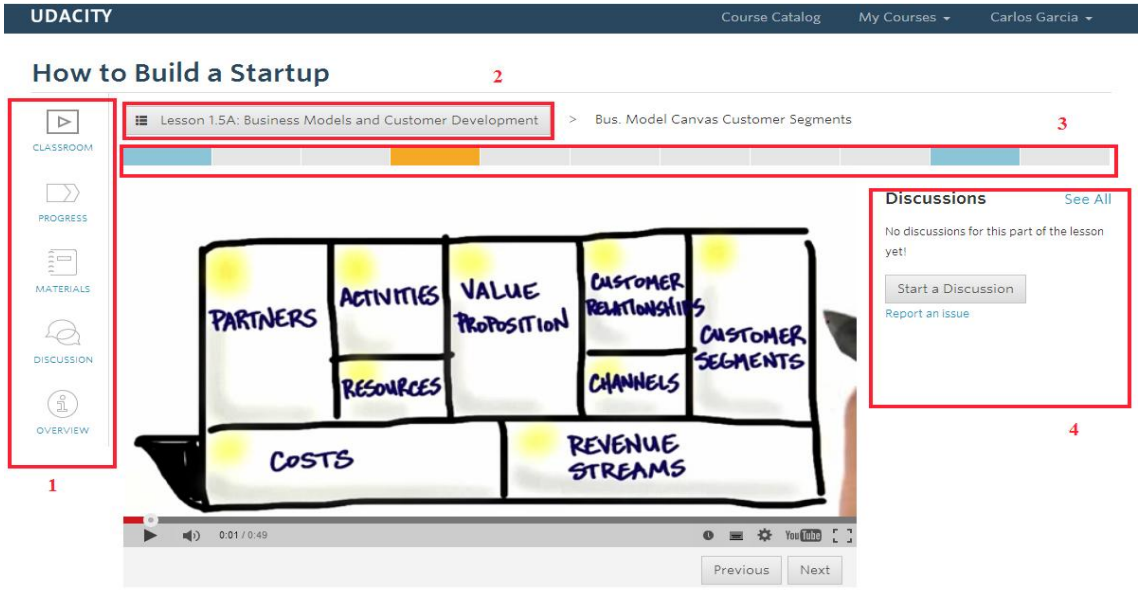


Figura 2.9 *Interfaz de un curso*.

A parte de la navegación por los contenidos del curso, podemos encontrar otra serie de funcionalidades que se suelen ver en las plataformas dedicadas a MOOCs. Por un lado, una sección donde encontramos el progreso que llevamos dentro del curso, donde nos aparecen marcadas las lecciones superadas y las que nos quedan por superar (1)(Figura 2.10). Una sección donde tenemos acceso a todo el contenido del curso a modo de apuntes del profesor (2)(Figura 2.10). También tendríamos una sección de discusiones a modo de foro con funcionalidades como: puntuar comentarios o discusiones, y el etiquetado y búsqueda por etiquetas (3)(Figura 2.10). Y por último una sección con un resumen informativo acerca del curso (cómo se supera, duración, nivel requerido, profesores participantes, syllabus…) (4)(Figura 2.10).

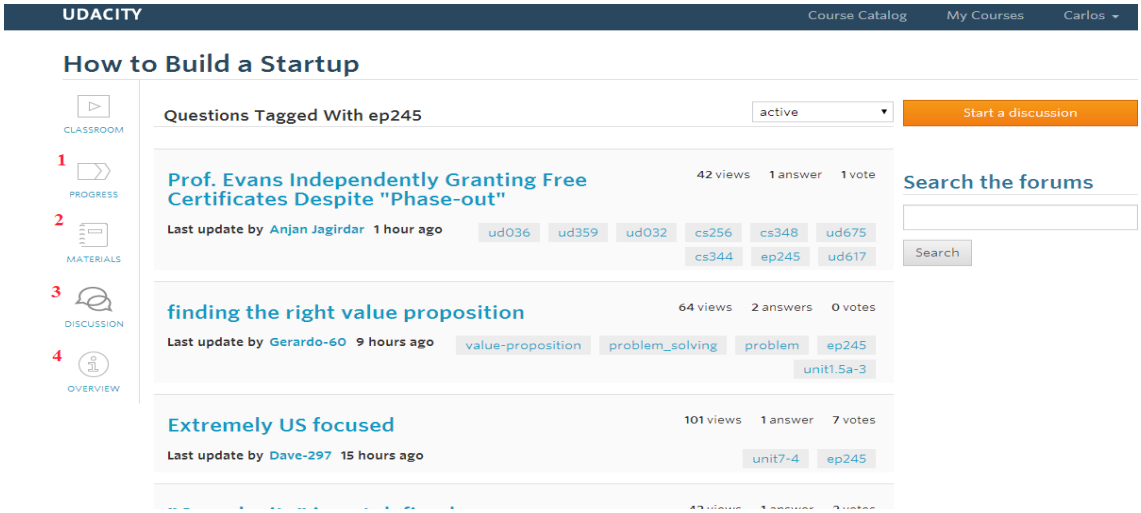


Figura 2.10 *Secciones de un curso*.

2.4 MOOCs: ¿Qué persiguen?

Como hemos podido apreciar, las plataformas que proveen MOOCs tratan de ofrecer una experiencia transparente y consistente mediante un diseño que se apoya en tres fundamentales principios [11]:

- **Simplicidad:** apoyan la estructura de la página sobre el contenido, dando protagonismo a este, ya que la claridad del contenido y la reducción de distracciones innecesarias son esenciales en una plataforma educativa.

- **Integración:** las herramientas son ofrecidas en la interfaz y están muy a mano para un acceso fácil a la información que necesites en cualquier momento. El progreso y la navegación a todos los contenidos del curso también se ofrecen en la interfaz para tener un rápido acceso a una visión de conjunto del curso. Además, cuentan con discusiones integradas dentro del contenido para ver qué opinan los demás usuarios de este de forma fácil y rápida.

- ***Responsiveness*:** debido a la gran demanda en las formas de consumir contenido en internet, las plataformas de MOOCs emplean tecnologías web para ofrecer una interfaz muy elástica, capaz de escalar y cambiar dependiendo del dispositivo y por tanto del tamaño de pantalla del navegador.

2.5 LCMSs: Moodle.

Durante los últimos años las plataformas de MOOCs han ocupado los principales titulares en cuanto a educación online, pero no podemos olvidar al que ha sido el gran dominador de los espacios educativos virtuales durante la última década, Moodle.

Moodle se trata de un sistema de gestión de cursos, también conocido como LCMS (*Learning Content Management* System), de distribución libre, para la creación entornos de aprendizaje en línea. La primera versión salió en el 2002 y actualmente se encuentra en la versión 2.6 con más de 21 millones de usuarios registrados, distribuidos en 46.000 sitios en todo el mundo y ha sido traducido a 91 idiomas.

El diseño de Moodle está basado en la idea de crear un ambiente centrado en el estudiante, es decir, el profesor al utilizar Moodle debe crear dicho ambiente. Este ambiente promueve que el estudiante construya su conocimiento en base a sus habilidades y conocimientos propios en lugar de publicar y transmitir la información que los estudiantes deban conocer.

Moodle se ha convertido en una herramienta de gran utilidad, ya que ha permitido generar cursos virtuales que facilitan la educación a distancia o espacios virtuales que complementen a la educación presencial.

Para empezar a trabajar con Moodle, primero es necesario instalarlo. Esta instalación requiere una plataforma que soporte PHP 5.2.8 o superior y alguna de las siguientes bases de datos: PostgreSQL 8.3 o superior, Oracle 10.2 o superior, MS SQL 2005 o superior.

Una vez instalada la plataforma Moodle, esta cuenta con un gran número de funcionalidades tanto en administración como desarrollo del curso [12].

Funcionalidades de administración.

Moodle presenta una gran variedad de funcionalidades respecto a la administración de los cursos de las que merece la pena destacar las siguientes:

- Un administrador definido durante la instalación.
- Exponer determinadas funciones de Moodle como servicios web basados en estándares.
- Capacidad multilenguaje.
- Añadir nuevos módulos de actividades a los ya instalados en Moodle.
- Elegir entre varios formatos de curso tales como semanal, por temas o el formato social, basado en debates.
- Personalización del sitio utilizando "temas" que redefinen los estilos, los colores del sitio, la tipografía, la presentación, la distribución, etc (Figura 2.11).

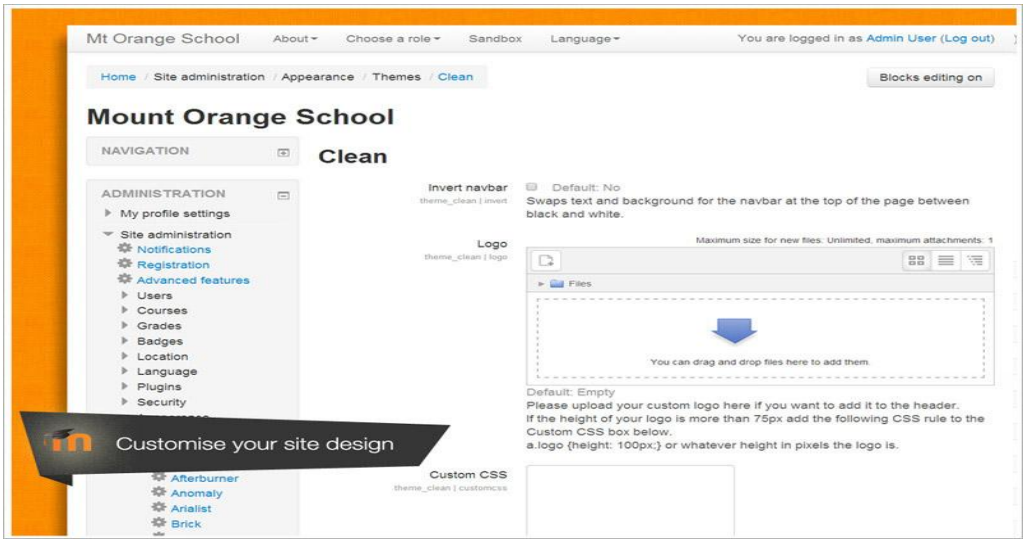


Figura 2.11 *Personalización de los temas*.

- La mayoría de las áreas para introducir texto (materiales, envío de mensajes a un foro, entradas en el diario, etc.) pueden editarse usando un editor HTML WYSIWYG integrado.

- Gestionar roles y permisos de usuarios (Figura 2.12).

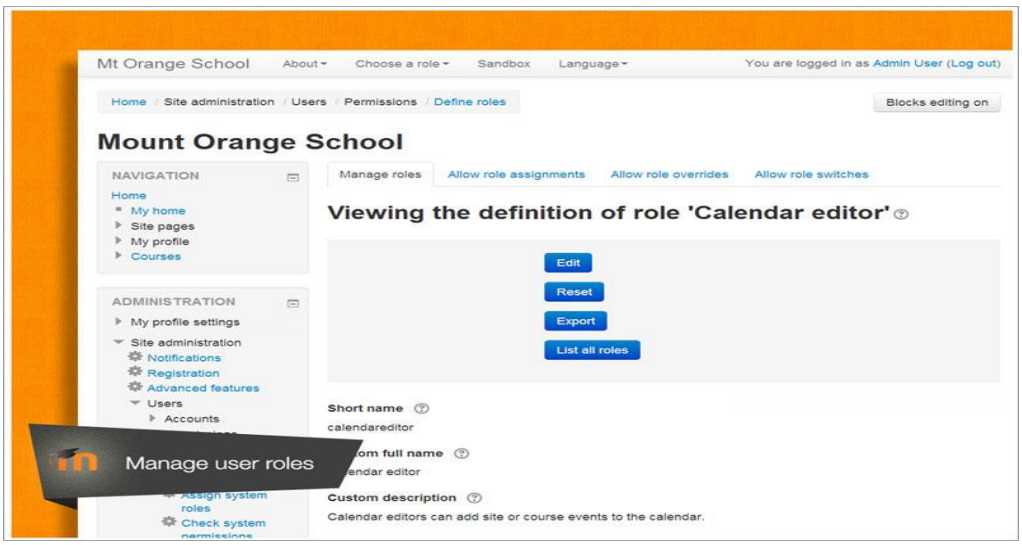


Figura 2.12 *Gestión de permisos*.

- Todas las calificaciones para los foros, diarios, cuestionarios y tareas pueden verse en una única página (y descargarse como un archivo con formato de hoja de cálculo).Se dispone de informes de actividad de cada estudiante, con gráficos y detalles sobre su paso por cada módulo así como también de una detallada "historia" de la participación de cada estudiante, incluyendo mensajes enviados, entradas en el diario, etc. en una sola página (Figura 2.13).

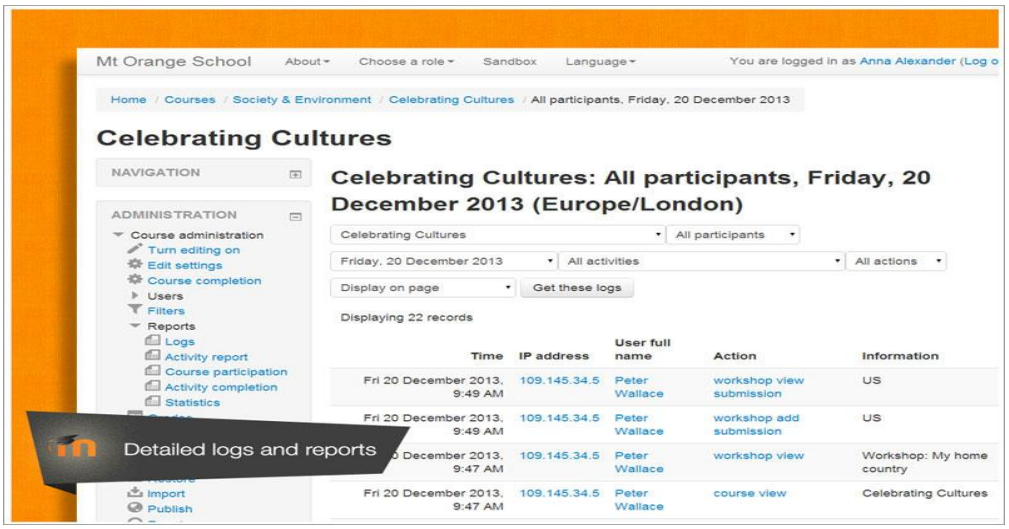


Figura 2.13 *Informes de actividad.*

Funcionalidades de desarrollo del curso.

Al igual que en la administración de un curso, Moodle tambiéncuenta con un amplio abanico de funcionalidades en el desarrollo de un curso. Entre otras, podemos encontrar:

- Fomentar la colaboración, estimulando la colaboración dirigida por el contenido (Figura 2.14).

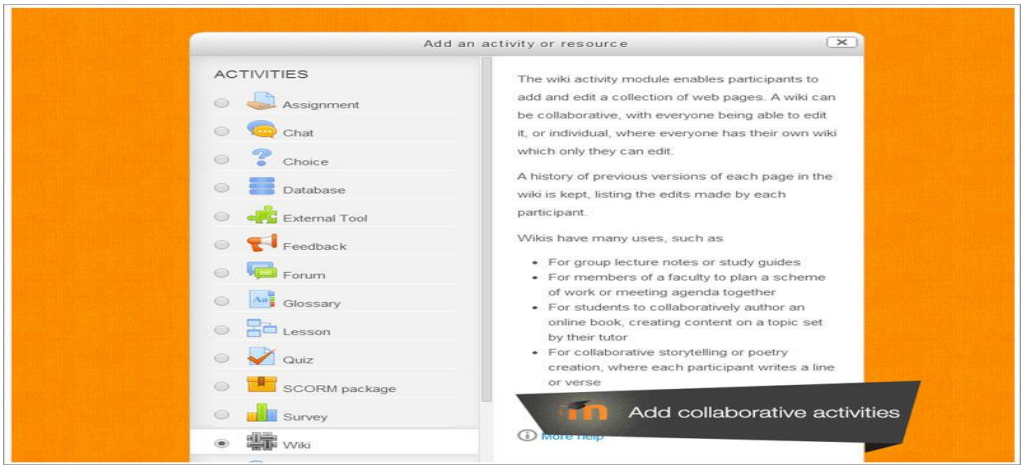


Figura 2.14 *Añadir funciones colaborativas***.**

- Integración de contenido multimedia (Figura 2.15).

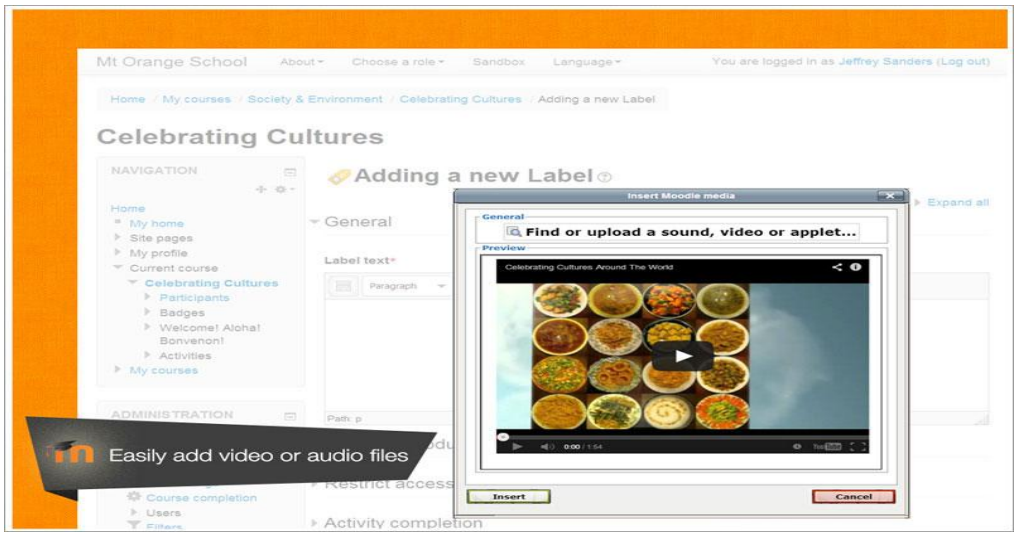


Figura 2.15 *Añadir contenido multimedia***.**

- Evaluación por pares y autoevaluación (Figura 2.16).

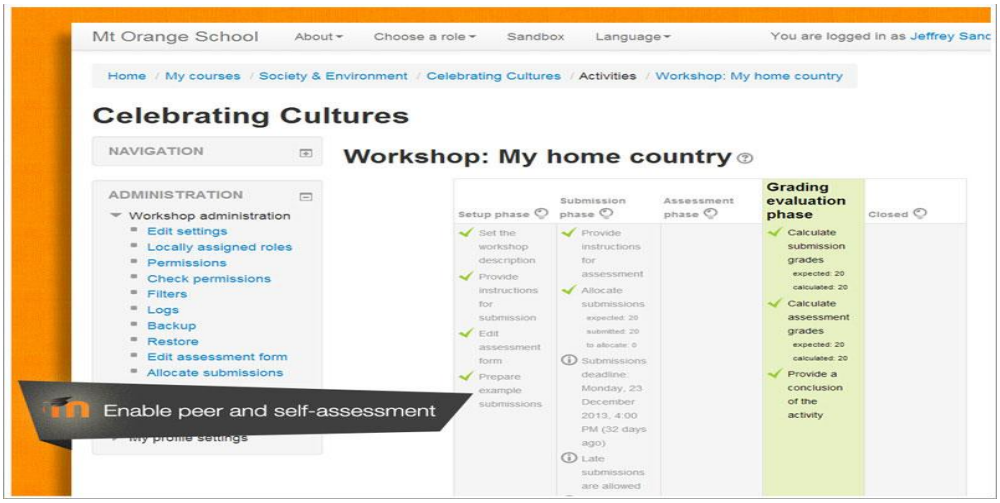


Figura 2.16 *Evaluación por pares*.

- Diferentes tipos de caminos de aprendizaje: clases guiadas por el instructor, semi-presenciales, online o autodidacta.
- Una de las características más atractivas de Moodle, que también aparece en otros gestores de contenido educativo, es la posibilidad de que los alumnos participen en la creación de glosarios, y en todas las lecciones se generan automáticamente enlaces a las palabras incluidas en estos.
- Capacidad de hacer anotaciones *in-line* sobre PDFs (Figura 2.17).

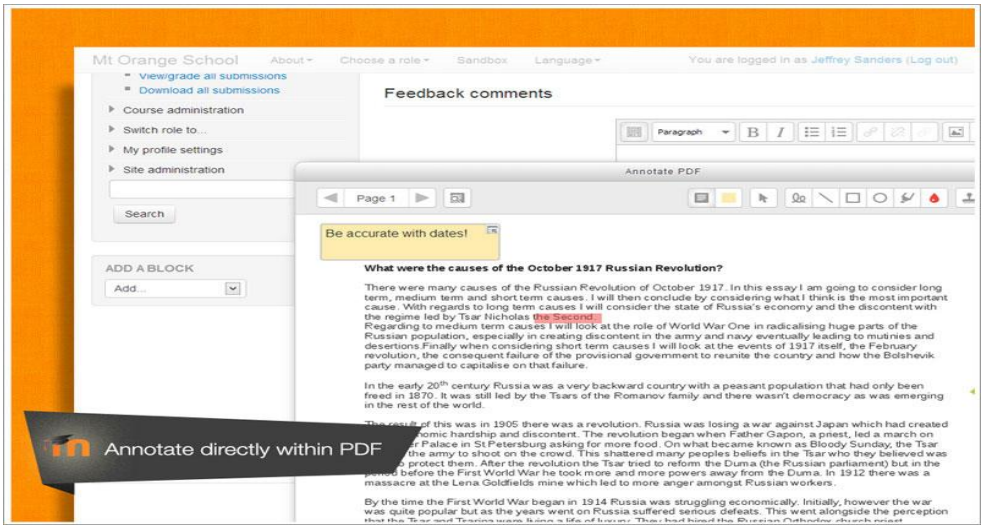


Figura 2.17 *Anotaciones in-line.*

- Gestión de grupos: compartir cursos y facilitar el trabajo en equipo (Figura 2.18).

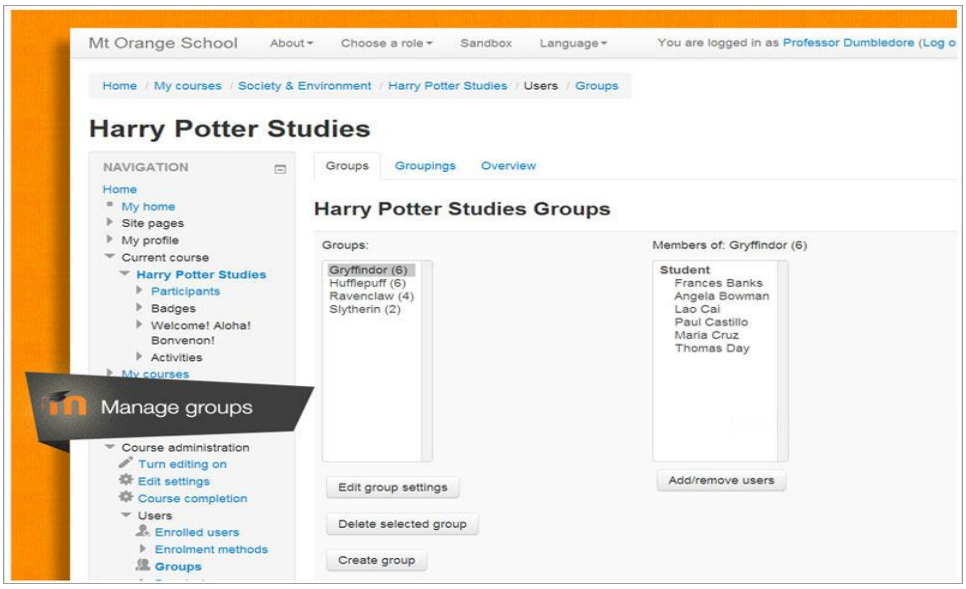


Figura 2.18 *Gestión de grupos de trabajo***.**

Moodle ha sido durante la última década la plataforma predominante en la enseñanza on-line. Y ha adquirido una larga vista de funcionalidades como hemos podido ver. Pero aun así, Moodle presenta una serie de carencias que han sido las más discutidas durante sus años de vida. De entre las cuales se destacan las siguientes:

- Algunas actividades pueden ser un poco mecánicas, dependiendo mucho del diseño de las instrucciones.
- Por estar basado en tecnología PHP, la configuración de un servidor con muchos usuarios debe ser cuidadosa para obtener el mejor desempeño.
- Falta mejorar su interfaz de una manera más sencilla.
- Hay desventajas asociadas a la seguridad, dependiendo en dónde se esté alojando la instalación de Moodle y cuáles sean las políticas de seguridad y la infraestructura tecnológica con la cual se cuente durante la instalación.
- La plataforma puede no ser relativamente fácil para muchos usuarios.

Moodle hace de golpe lo que hoy en día hacen muchas aplicaciones por separado y no está claro que papel desempeñará, por tanto, Moodle en el futuro. Pero precisamente la aparición de nuevas herramientas con el paso del tiempo ha hecho que Moodle deje de

estar en el centro para pasar a formar parte de todo un diverso ecosistema de herramientas interrelacionadas donde cada una hace lo que mejor sabe hacer: Moodle gestiona los cursos y otros sistemas se encargan del Portafolio, de la gestión de archivos, de multimedia, etc.

Resumiendo, probablemente Moodle y su capacidad de crear espacios para el aprendizaje encontrará su lugar formando parte de un ecosistema de diferentes aplicaciones donde cada una hará bien aquello por lo que ha sido pensada.

2.6 Tecnologías de desarrollo.

El objetivo de este proyecto como de cualquier otro proyecto de desarrollo de software, es el de obtener un software de calidad. Pero esta empresa lejos de estar al alcance de un ingeniero informático, se vislumbra casi utópica dentro de la vorágine que supone el desarrollo del software en la actualidad. Pues la calidad, ya sea de software o de cualquier otro producto, está siempre asociada al coste de este. Y la suposición que subyace aquí es que la calidad es negociable, que disminuyendo la calidad se gana en coste y velocidad. Esto nos lleva a pensar que antes que invertir tiempo en seguir buenas prácticas en el desarrollo del software es preferible añadir más funcionalidades.

Este enfoque a la hora de desarrollar software, muchas veces propiciado por la presión que el *time-to-market* ejerce sobre el proyecto, hace que el software acabe saliendo más caro de lo que se pretendía.

Para evitar estos problemas y conseguir entregar un software de calidad es fundamental llevar a cabo integración continua durante el desarrollo. No llevar a cabo estas buenas prácticas te colocan en una posición de ceguera en cuanto a que no sabes en que momento del desarrollo te encuentras, qué funciona o qué no, mientras que seguirlas te ayudan a desarrollar un software de calidad sin incrementar el tiempo de desarrollo.

Una integración continua es esencial, y para llevarla a cabo es necesario echar mano de las tecnologías existentes para este fin. Estas tecnologías facilitan la vida de los desarrolladores y pueden marcar la diferencia entre un proyecto exitoso y uno que no lo es. A continuación hablaremos de este tipo de tecnologías.

2.6.1 CoffeeScript: Lenguaje de programación.

El código se compila directamente en el equivalente JavaScript sin interpretación de código en tiempo de ejecución.

JavaScript tiene muchas singularidades que hacen cometer errores a los desarrolladores menos expertos. Por el contrario, CoffeeScript evita estos errores comunes exponiendo solo las "buenas" características de JavaScript.Las principales características de CoffeeScript son:

- Puedes usar cualquier librería de JavaScript desde CoffeScript y viceversa.
- El código compilado es claro y fácilmente leíble.
- El código compilado funciona en todos los tiempos de ejecución de JavaScript y suele ser tan rápido o más que su equivalente JavaScript.
- Menos código.
- Existen clases.
- Toma espacios en blanco en consideración.

Una de las grandes falacias a desmentir sobre CoffeeScript es que no necesitas saber JavaScript. Sí es necesario conocer JavaScript, pues los errores en tiempo de ejecución necesitan conocimiento de JavaScript para saber que está ocurriendo en el programa a la hora de depurar.

Aun teniendo que entender JavaScript para depurar, CoffeeScript supone una gran ventaja, ya que su fácil entendimiento te ahorra mucho tiempo cuando tienes que depurar un gran número de líneas de código.

Por otra parte están sus desventajas, de las que cabe destacar que utilizar CoffeeScript supone añadir una compilación más, lo que dificulta la integración en servidores con compilaciones automáticas. Otra desventaja reside en que la comunidad de CoffeScript es aún pequeña.

La sintáxis de CoffeeScript está inspirada en Ruby, lo que hace que prime la convención sobre configuración, además de eliminar caracteres como llaves, comas o puntos y comas para eliminar ruido a la hora de leer el código.

2.6.2 Node.js: Entorno de programación.

Node.js es un entorno de programación de la capa del servidor basado en el lenguaje de programación Javascript, con una arquitectura orientada a eventos, y basado en el motor Javascript V8 de Google Chrome.

Node.js utiliza como paradigma de programación la programación orientada a eventos, donde el flujo de acciones está determinado por la ocurrencia de eventos. Para ello, los programadores registran *callbacks* que serán usados como controladores de la ocurrencia de un evento en el que estén interesados, es decir, el sistema invocará estos controladores cuando el evento al que están asociados tenga lugar [13].

Todo esto se apoya sobre el *Event Loop*. El *Event Loop* es la parte fundamental de Node.js y se trata de un único hilo de ejecución que ejecuta una única cosa en cada momento, por lo cual no hay concurrencia, es decir, no hay problemas de condiciones de carrera ni cerrojos. Esto hace la programación del servidor bastante más simple que con otros lenguajes.

Además, es no-bloqueante, todas las operaciones de entrada/salida quedan registradas como eventos. Una vez Node.js termine con los trabajos y haya un evento registrado en la cola de eventos, Node.js recuperará y ejecutará el *callback* de este. Node.js siempre está trabajando, no deja de atender peticiones, siempre está ejecutando algo.

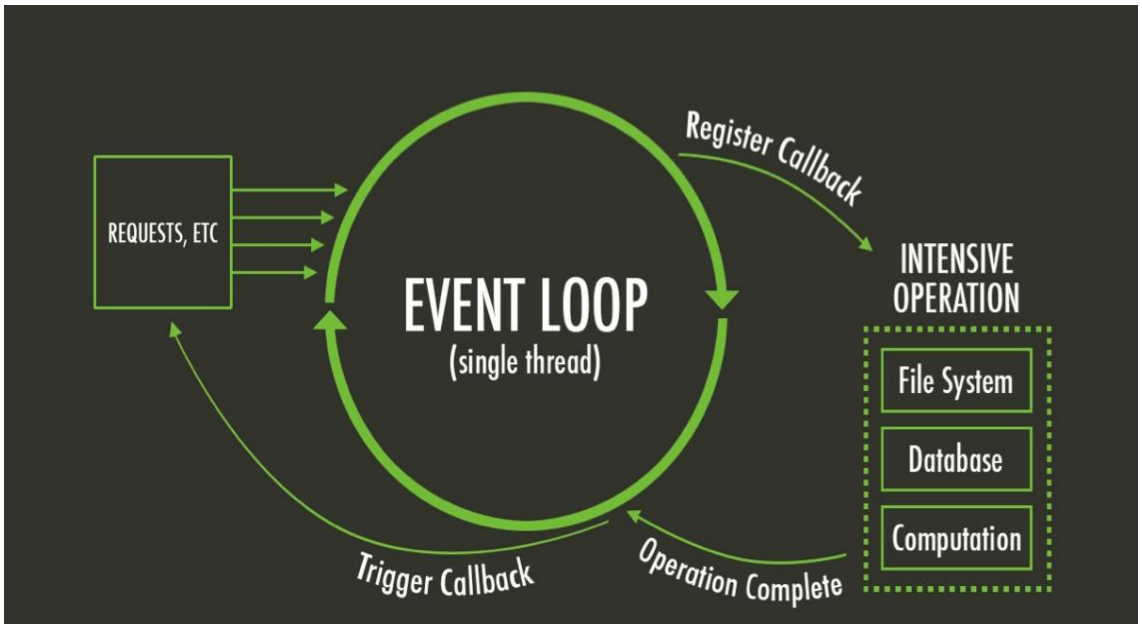


Figura 2.19 *Flujo de tareas del Event Loop.*

La complicación de Node.js aparece cuando es necesario anidar *callbacks*, lo que da lugar a un código difícil de leer y mantener. Este anidamiento es necesario cuando se quiere ejecutar algo en serie con una llamada asíncrona (entrada/salida), sino correrá el peligro de ejecutarse antes de que se ejecute el *callback,* cuando lo que queremos es que el *callback* se ejecute primero. Esto se debe a que Node.js registrará el evento y seguirá ejecutando ya que es no-bloqueante.

Es aconsejable seguir dos principios al programar en node.js, porque también hay la posibilidad de, aun utilizando *callbacks* (asincronismo), el *throughput* del servidor quede acotado por un solo *callback*:

1. Una vez se haya hecho el *setup* del servidor hay que realizar todas las acciones de modo que sean orientadas a eventos.
2. Si Node.js va a procesar algo durante que dure un largo tiempo, habrá que considerar delegarlo a "*web workers*".

Debido a que el modelo de programación de Node.js está basado en un único hilo de ejecución, Node.js obtiene una gran escalabilidad en comparación con el tradicional modelo bloqueante donde para escalar necesitas múltiples procesos e hilos. Por ello Node.js está pensado para aplicaciones donde necesitas hacer muchas cosas al mismo tiempo, sobre todo de entrada/salida a la vez. Y es especialmente bueno para aplicaciones *realtime*. En cambio, si lo que necesitas es trabajo intenso de CPU, Node.js no supone ninguna ventaja [14].

Otras ventajas que trae Node.js consigo son la utilización del mismo lenguaje tanto en cliente como servidor, una buena gestión de paquetes gracias a NPM (*Node Package Manager)* y la gran comunidad de desarrolladores detrás de Node.js, la cual continúa creciendo.

2.6.3 Neo4j: Base de datos de grafos.

Neo4j es una base de datos NoSQL basada en grafos, la cual se caracteriza por ser robusta, escalable, de alto rendimiento, y que gestiona eficientemente datos semi-estructurados y orientados a redes [15]. Con un lenguaje de consultas propio denominado *Cypher*.

Datos orientados a redes: mientras que la mayoría de bases de datos tienen un modelo relacional basado en tablas, columnas, y filas, Neo4j se basa en nodos, relaciones y propiedades para formar una extensa red de información (Figura 2.20). Una relación conecta dos nodos, tiene un tipo definido y puede ser opcionalmente dirigida. Las propiedades son un par clave-valor que van ligadas tanto a los nodos como a las relaciones.

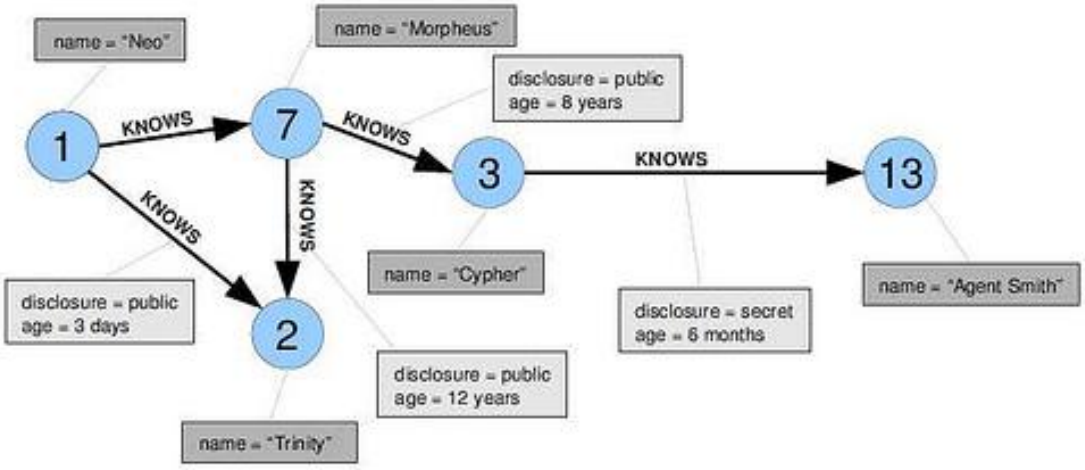


Figura 2.20 *Ejemplo de red en Neo4j*

Datos semi-estructurados: los datos semi-estructurados se caracterizan por tener, por un lado atributos obligatorios, y por otro, atributos opcionales. Y por tanto, poseen una estructura muy dinámica con datos que tienen un grado de diferenciación tan elevado que son difíciles de manejar con un esquema de bases de datos relacionales, pero que por el contrario, pueden ser fácilmente representados en el modelo de Neo4j.

Escalable y de alto rendimiento: Neo4j fue pensada desde el primer momento para ser escalable y de alto rendimiento. Para hacernos una idea del poder de Neo4j hay que saber que cada salto de un nodo a otro a través de una relación es equivalente a un *join* en una base de datos relacional. A continuación, mostraremos una comparativa de rendimiento entre las bases de datos relacionales y Neo4j para una red social con 1.000.000 de personas con aproximadamente 50 amigos cada una. Los resultados demuestran como una base de datos de grafos es la mejor elección para datos conectados (Tabla 2.1).

Profundidad	RDBMS tiempo ejecución (s)	Neo4j tiempo ejecución(s)	Resultados devueltos
2	0.016	0.01	~2500
3	30.267	0.168	~110.000
4	1543.505	1.359	~600.000
5	No finalizó	2.132	~800.000

Tabla 2.1 *Benchmarking Neo4j.*

Hablando en términos de funcionalidades, Neo4j presenta las siguientes características:

- Transacciones ACID.
- Alta disponibilidad.
- Capacidad de escalar hasta miles de millones de nodos y relaciones.
- Alta velocidad de consulta por recorrido (*traversal*).

- Lenguaje de consultas declarativo (Cypher).

En consecuencia con las características de Neo4j, esta base de datos de grafos presenta grandes ventajas a la hora de desarrollar software. Basta darse cuenta, de que si tu aplicación maneja un dominio que contiene datos, que por naturaleza vienen estructurados en redes, el modelo de Neo4j es compatible con ese modelo de forma nativa. Al igual ocurre con datos semi-estructurados como es la Web. En este caso, Neo4j también es capaz de lidiar más fácilmente con este tipo de dominio debido a que su modelo está mucho más desacoplado que el de una RDBMS. Esto ayuda una vez que la aplicación esté en producción. Además, la flexibilidad de Neo4j permite refactorizar los modelos de datos rápidamente, lo que favorece las metodologías ágiles con iteraciones cortas. Todas las características de Neo4j combinadas, hacen de Neo4j una base de datos que ayuda a acortar tiempos de desarrollo, reducir costes de mantenimiento y aumentar el rendimiento.

Modelos de datos.

Neo4j es una base de datos de grafos, y por lo tanto cuenta con una serie de elementos que sirven para conformar el modelo de datos con los que no cuentan las RDBMS. Los elementos que encontraremos son: *nodes*, *relationships*, *properties*, *labels*, *paths*, *traversal* y *schemas* mediante *indexes* y *constraints*. A continuación explicaremos en detalle las funciones que cumplen cada uno de ellos [16](Figura 2.21).

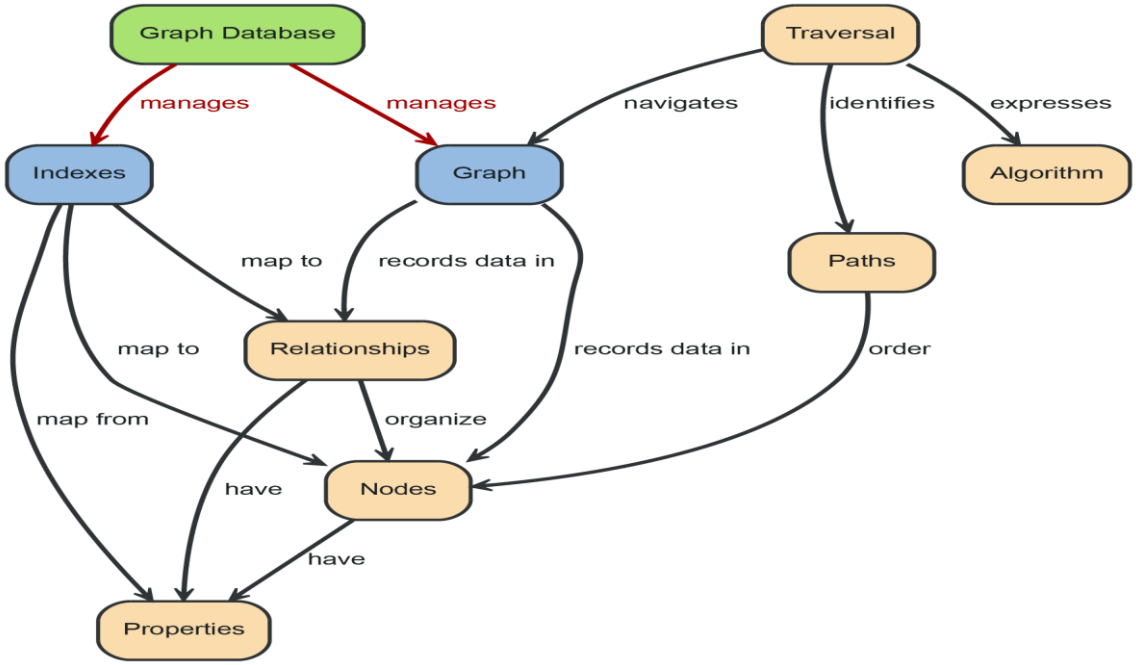


Figura 2.21 *Descripción de Neo4j a modo de grafo.*

Nodes: Son las unidades fundamentales de Neo4j y suelen representar *entidades* del dominio. Pueden contener propiedades y ser etiquetados con una o más etiquetas.

Relationships: establecen conexiones entre nodos, estableciendo así valor semántico dentro del dominio y posibilitando las búsquedas entre datos relacionados. Al igual que los nodos, pueden tener asociadas propiedades. Las relaciones tienen un nodo de comienzo y otro de fin, son dirigidas, y pueden ser recorridas en ambos sentidos, por lo que no es necesario establecer relaciones duplicadas para cada sentido. Además, las relaciones poseen un tipo que hace que el recorrido por el grafo sea más eficiente.

Properties: son pares clave-valor, donde la clave es un *String* mientras que los valores pueden ser una primitiva o un *array* de primitivas. *Null* no es un valor válido pero puede ser modelado como la ausencia de clave.

Labels: permite agrupar nodos en conjuntos de nodos para que las consultas sobre la base de datos solo tengan que trabajar sobre esos conjuntos en lugar de con todo el grafo, haciendo las consultas más eficientes. Estas etiquetas pueden ser añadidas y borradas durante el tiempo de ejecución lo que posibilita, por ejemplo, marcar estados temporales asociados a los nodos. Para los nombre de etiquetas, cualquier *string unicode* no vacío pude ser utilizado.

Paths: los caminos son nodos con relaciones conectadas devueltos como resultado de una consulta que conlleva o no un recorrido en el grafo. Es decir, pueden llegar a tener una longitud de cero lo que equivaldría a un nodo.

Traversal: *traversing* un grafo significa recorrer los nodos a través de sus relaciones de acuerdo a unas reglas. Neo4j viene con una API basada en *callbacks* que permite especificar dichas reglas como por ejemplo *breath-first search* o *depth-first search*.

Schema: en Neo4j el *schema* es opcional, pero la introducción de un esquema es esencial si quieres ganar rendimiento y beneficios en el modelado de datos. Esto te permite trabajar sin *schema* hasta que quieras utilizar los beneficios de tener uno.

Para la creación de un *schema* que guie el modelado de datos en tu dominio, Neo4j cuenta por un lado con ***indexes*** que aumentan el rendimiento permitiéndote aumentar la búsqueda de nodos por sus propiedades indexadas. Por otro lado, tienes las ***constraints*** que te permiten mantener los datos limpios, ya que puedes especificar cualquier tipo de cambio en los datos no deseado y así dichos cambios serán denegados por el sistema.

Cypher: Lenguaje de consultas.

Cypher es un lenguaje de consultas declarativo que permite consultas expresivas y eficientes. Está diseñado para ser un lenguaje de consulta que se adapte a la forma que tienen los humanos de entender la sintaxis. Por ello, las construcciones del lenguaje *Cypher* están basadas en prosa inglesa y una limpia iconografía, lo cual hace de *Cypher* un lenguaje auto-explicatorio. *Cypher* se centra en buscar la claridad de expresar **qué** se devuelve con la consulta, en lugar **cómo** se devuelve.

El lenguaje *Cypher* se compone de varias cláusulas:

- START: puntos de comienzo de búsqueda, obtenidos mediante índices o IDs de elementos.
- MATCH: patrón de coincidencia para la búsqueda que comience a partir de START.
- WHERE: criterio de filtrado.
- RETURN: resultados que se devuelven.
- CREATE: crea nodos y relaciones.
- DELETE: borra nodos, relaciones y propiedades.
- SET: asigna valores a las propiedades.
- FOREACH: realiza una actualización de datos por cada elemento de una lista.
- WITH: dividir una consulta en múltiples partes.

Para finalizar, hay que destacar que para extraer o modificar información, *Cypher* utiliza la expresividad del lenguaje como patrón de búsqueda, es decir, se dedica a buscar coincidencias con dicho patrón (relaciones y nodos) dentro del grafo. Los patrones que podemos encontrar son:

- Nodos relacionados: (a) --> (b).
- Etiquetas: (a:User:Admin)-->b.
- Relaciones: a-[r:TYPE1|TYPE2]->b
- Propiedades: (a: User {property1: “value”, property2: “value”}).

2.7 ¿De dónde partimos?

El módulo de cursos queda englobado dentro del sistema CloudRoom [17]. Una plataforma MOOC con una arquitectura ajustada a las últimas convenciones en *Cloud Computing* y basada en una arquitectura orientada a servicios (Figura 2.22).

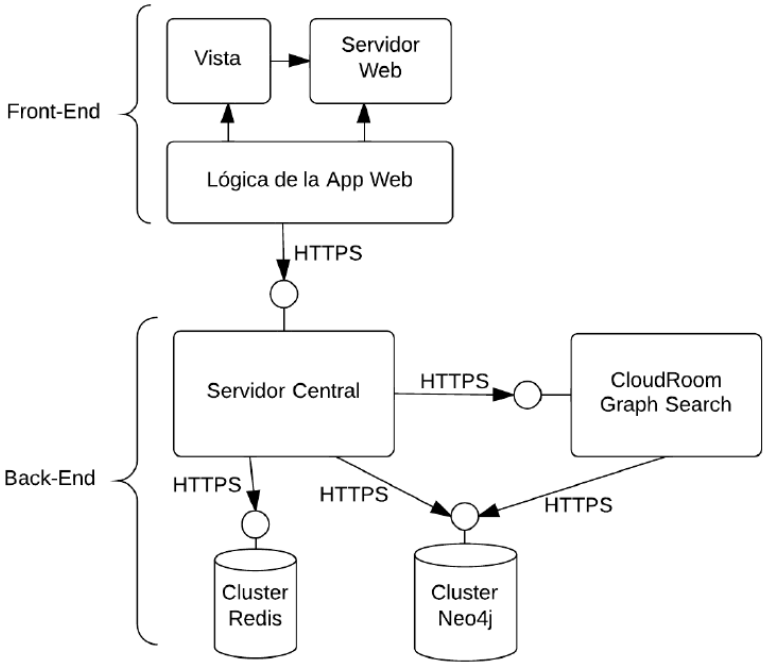


Figura 2.22 *Arquitectura de CloudRoom.*

En dicha plataforma, se pretende dar cabida a *Massive Online Open Courses* (MOOCs), fomentando así, un aprendizaje ubicuo, en cualquier lugar y cualquier momento, a través de todo tipo de clientes móviles. Además, se busca un aprendizaje social mediante la interacción entre alumnos, profesores e instituciones a través de la plataforma (Figura 2.23).

Esta plataforma ha sido creada por alumnos [18], para alumnos, y busca la integración futura de técnicas de gamificación y *Linked Data* para favorecer el aprendizaje.

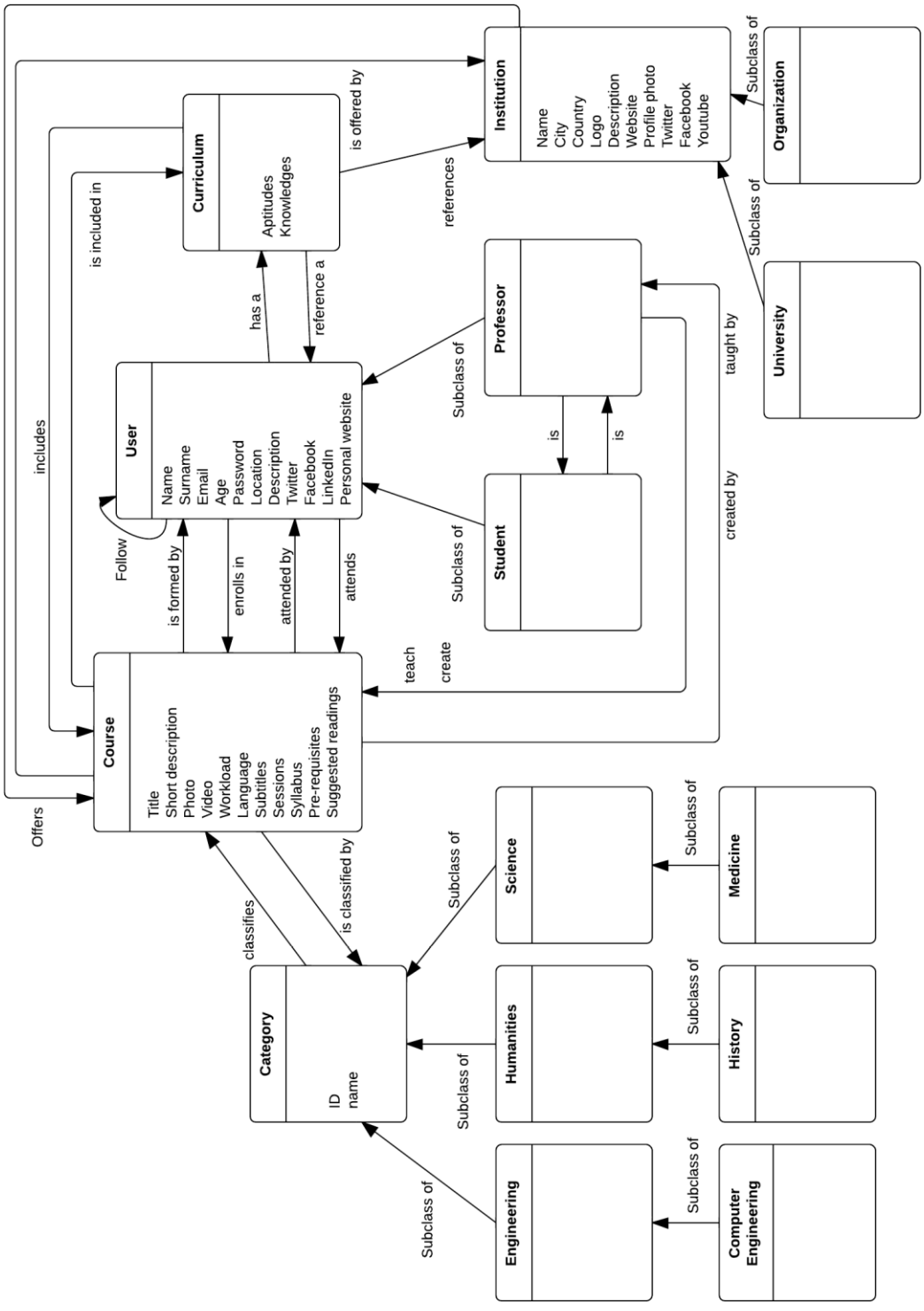


Figura 2.23 *Modelo del dominio de CloudRoom.*

REQUISITOS

"Escribir software que satisfaga plenamente una especificación es como caminar sobre el agua. En ambos casos, lo primero es fácil si lo segundo está congelado, y casi imposible si es líquido."

Anónimo

3 Planteamiento del problema.

La propuesta que se nos plantea se trata de realizar el módulo que dé cabida a las funcionalidades que requieren los usuarios de nuestra plataforma. Para ello, se extraerán y priorizarán una serie de requisitos. Estos requisitos se presentan como requisitos funcionales y requisitos no-funcionales.

Los requisitos funcionales definirán las *features* a implementar, es decir, funcionalidad del sistema que abordará necesidades del usuario. Por otro lado, están los requisitos no-funcionales, los cuales se presentan en forma de atributos de calidad del sistema (interoperabilidad, disponibilidad, rendimiento, usabilidad…). Estos requisitos no-funcionales guiarán el desarrollo del módulo puesto que los requisitos funcionales serán validados contra los requisitos no-funcionales.

En nuestro caso, muchos de los requisitos no-funcionales vienen dados a modo de restricciones previas. Estas restricciones previas suelen ser objetivos de negocio (*outsourcing*, *time-to-market*, reutilización de una arquitectura software para establecer una familia de productos software…), pero en este caso, y al tratarse del diseño de un módulo englobado dentro de un sistema cuya arquitectura software ya ha sido diseñada, las restricciones vendrán dadas por esas decisiones arquitectónicas ya tomadas.

El módulo a diseñar se trata de una API REST para favorecer la interoperabilidad entre los diferentes dispositivos que albergarán el cliente de CloudRoom (móviles, tabletas o navegadores web). Este patrón también favorece la modificabilidad desacoplando el *front-end* del *back-end*. Además de la interoperabilidad y modificabilidad, que ya vienen dadas por el patrón de diseño, existen otros atributos de calidad para cada requisito que deberán ser conseguidos durante el diseño mediante el empleo de diferentes tácticas dependiendo del atributo de calidad que se pretenda conseguir.

Para extraer los requisitos se elaborará una lista con los casos de uso más prioritarios. Para estos casos de uso se establecerán una serie de escenarios donde se describen los requisitos no-funcionales, y luego extraeremos los requisitos funcionales de cada uno de estos casos de uso.

3.1 Análisis de Casos de Uso.

Como parte de una pequeña fase de planificación se han identificado requisitos a partir de la metodología de casos de uso. Estos casos se explicarán a continuación y están sujetos a cambios en los requisitos.

3.1.1 Lista de Casos de Uso.

A continuación se enumeran y explican brevemente los casos de uso que se han identificado (Tabla 3.1):

ID	Nombre	Actor	Prioridad	Complejidad	Frecuencia
1	Ver un listado de los anuncios del curso	Alumno	Alta	Baja	2-3 veces por sesión
2	Filtrar los anuncios del curso	Alumno	Alta	Baja	2-3 veces por sesión
3	Navegar por el contenido del curso	Alumno	Alta	Media	1 vez cada 3 min por sesión
4	Ver video de la lección	Alumno	Alta	Alta	5-6 veces por sesión
5	Ver apuntes de la lección	Alumno	Alta	Baja	5-6 veces por sesión
6	Realizar test evaluativo.	Alumno	Alta	Media	2-3 veces por sesión
7	Realizar quiz.	Alumno	Alta	Media	3-4 veces por sesión
8	Ver un resumen del progreso en el curso	Alumno	Media	Media	1 vez por sesión
9	Ver un listado de las tareas asignadas	Alumno	Media	Baja	1 vez por sesión
10	Realizar evaluación por pares.	Alumno	Baja	Alta	1 vez por mes
11	Subir ejercicios resueltos	Alumno	Media	Media	1 vez cada 15 días
12	Consultar estadísticas curso	Profesor	Baja	Media	1 vez / semana
13	Publicar anuncios en el curso	Profesor	Media	Media	1 vez / semana
14	Administrar contenidos	Profesor	Media	Media	1 vez / semana

Tabla 3.1 *Listado de Casos de Uso.*

3.1.2 Diagrama de Casos de Uso.

Siguiendo una notación UML hemos construido el diagrama para delimitar la funcionalidad del módulo de cursos (Figura 3.1).

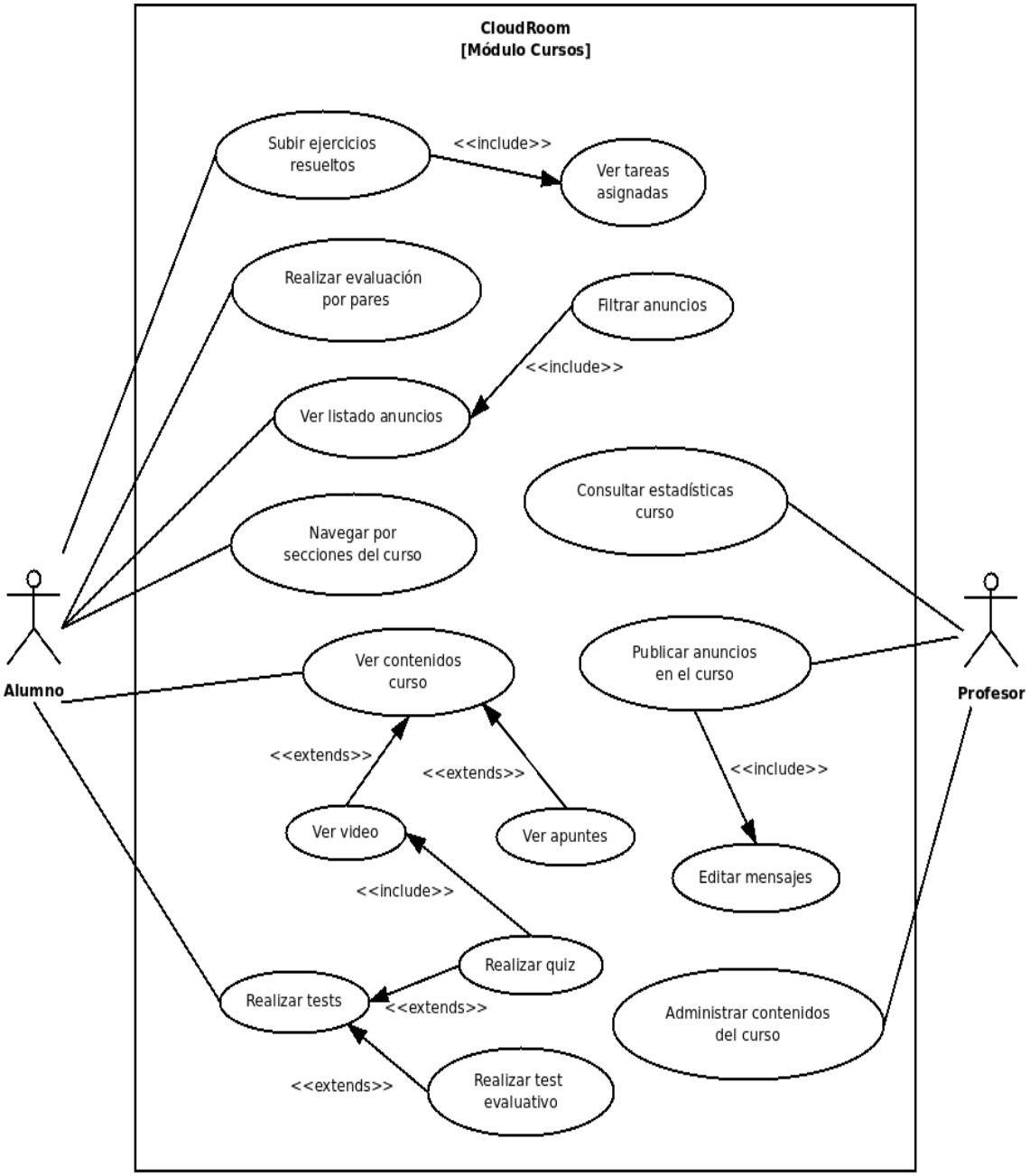


Figura 3.1 *Diagrama de Casos de Uso.*

3.1.3 Descripción de Casos de Uso: Extracción de requisitos.

A continuación pasamos a enumerar los casos de uso que hemos identificado como más prioritarios. Estos casos de uso han sido descritos en profundidad en esta sección y se han identificado los requisitos funcionales, y los requisitos no-funcionales a partir de escenarios de atributos de calidad, para cada uno de ellos.

Caso de Uso 1: Ver un listado de los anuncios del curso.

El alumno podrá ver todos los anuncios relativos al curso en el que está registrado. Estos anuncios pueden ser sobre discusiones del foro, ejercicios o anuncios realizados por el profesor, los ejercicios o la institución a la que pertenece el curso. Los anuncios aparecerán listados y ordenados cronológicamente con los más actuales al principio. Esto permitirá al alumno estar al tanto de cualquier cambio en el curso que pueda afectarle (Tabla 1.2).

Caso de Uso ID:	1		
Caso de Uso Nombre:	Ver un listado de los anuncios del curso		
Creado por:	Carlos García	Actualizado por:	
Fecha creación:	14/04/2014	Fecha actualización:	
Descripción:	El alumno entra en el curso en el cual se ha registrado previamente y visualiza los anuncios relacionados con el curso. El sistema le mostrará los últimos 25 anuncios. En caso de querer ver anuncios más antiguos, tendrá la posibilidad de solicitarle más al sistema de 10 en 10.		
Actores:	Alumno		
Disparador:	El alumno desea conocer los anuncios del curso.		
Precondiciones:	1. El alumno esta logueado en el sistema. 2. El alumno ha accedido a un curso en el que está registrado.		
Postcondiciones:	1. Se muestran los 25 anuncios más actuales. 2. Se muestran los primeros 25 anuncios más el resto que haya solicitado de 10 en 10.		

Escenario normal:	1.0.1 El alumno accede al curso en el que está registrado. 1.0.2 El sistema muestra en pantalla los 25 anuncios más actuales del curso. 1.0.3 El alumno sale del curso.
Escenarios alternativos:	Escenario alternativo 1 1.1.1 El alumno accede al curso en el que está registrado. 1.1.2 El sistema muestra en pantalla los 25 anuncios más actuales del curso. 1.1.3 El alumno desea ver más anuncios. 1.1.4 El sistema muestra los 10 siguientes anuncios más actuales. 1.1.5 El alumno repite el paso 1.1.3 tantas veces como quiera. 1.1.6 El alumno sale del curso.
Excepciones:	1.0.E.1 El alumno no está registrado en el curso. No podrá acceder a este y se le comunicará con un mensaje en pantalla. 1.0.E.2 No quedan más anuncios por mostrar cuando el alumno pide más, así que el sistema muestra un mensaje de advertencia.
Includes:	
Prioridad:	Alta: Estar al tanto de los anuncios permite al alumno orientarse dentro del curso y evitar su abandono por falta de información.
Frecuencia de uso:	2-3 veces cada vez que accede al curso
Complejidad:	Baja

Tabla 1.2 *Caso de Uso 1: Ver listado de anuncios del curso.*

Requisitos funcionales

- Como alumno quiero ver un listado de todos los anuncios relativos al curso ordenados cronológicamente empezando por los más actuales.
- Como alumno quiero obtener anuncios más antiguos además de los que ya estoy viendo.

Requisitos no-funcionales.

Rendimiento.

Cargar los anuncios de manera fluida es primordial para evitar que la experiencia de usuario se vea disminuida (Figura 3.2)(Figura 3.3).

Seguridad.

Es muy importante que el back-end de CloudRoom compruebe los campos del html que se cargará como anuncio para evitar que carguen *scripts* dañinos.

Figura 3.2 *Caso Uso 1: Escenario de Rendimiento 1***.**

Figura 3.3 *Caso Uso 2: Escenario de Rendimiento 2.*

Caso de Uso 2: Filtrar anuncios sobre el curso.

El alumno tendrá la posibilidad de filtrar los anuncios por cada una de las categorías (discusiones que sigue en el foro, anuncios profesor, anuncios institución) y así facilitar la búsqueda de los anuncios que le interesen. Esto permitirá al alumno estar al tanto de todos los hitos que se vayan produciendo en el curso y de este modo se le facilitará la orientación dentro del mismo (Tabla 3.3).

Caso de Uso ID:	2		
Caso de Uso Nombre:	Filtrar anuncios sobre el curso.		
Creado por:	Carlos García	Actualizado por:	
Fecha creación:	14/04/2014	Fecha actualización:	

Descripción:	El alumno entra en el curso en el cual se ha registrado previamente y visualiza los anuncios relacionados con el curso. Para no perder tiempo, el alumno decide buscar aquellos anuncios relacionados con los ejercicios, así que filtra por la categoría de ejercicios.
Actores:	Alumno
Disparador:	El alumno desea conocer los anuncios del curso.
Precondiciones:	1. El alumno esta logueado en el sistema. 2. El alumno ha accedido a un curso en el que está registrado.
Postcondiciones:	1. Solo se muestran los anuncios de la categoría seleccionada.
Escenario normal:	2.0.1 El alumno accede al curso en el que está registrado. 2.0.2 El sistema muestra en pantalla los anuncios del curso. 2.0.3 El alumno pulsa el botón que identifica la categoría de ejercicios por la cual desea filtrar los anuncios. 2.0.4 El sistema muestra solamente los anuncios de ejercicios escogidos por el alumno. 2.0.5 El alumno sale del curso.
Escenarios alternativos:	Escenario alternativo 1,2,3 2.1.1 El alumno accede al curso en el que está registrado. 2.1.2 El sistema muestra en pantalla los anuncios del curso. 2.1.3 El alumno pulsa el botón que identifica la categoría de anuncios sobre ejercicios/anuncios profesor/anuncios

	instituciones por la cual desea filtrar los anuncios. 2.1.4 El sistema muestra solamente los anuncios de ejercicios/anuncios profesor/anuncios instituciones escogidos por el alumno. 2.1.5 El alumno sale del curso
Excepciones:	2.0.E.1 El alumno no está registrado en el curso. No podrá acceder a este y se le comunicará con un mensaje en pantalla.
Includes:	
Prioridad:	Alta: Estar al tanto de los anuncios permite al alumno orientarse dentro del curso y evitar su abandono por falta de información.
Frecuencia de uso:	2-3 veces cada vez que accede al curso
Complejidad:	Baja

Tabla 3.3 *Caso de Uso 2: Filtrar anuncios sobre el curso.*

Requisitos funcionales.

- Como alumno quiero filtrar los anuncios por ejercicios.
- Como alumno quiero filtrar los anuncios por anuncios del profesor.
- Como alumno quiero filtrar los anuncios por anuncios de instituciones.

Requisitos no-funcionales.

Rendimiento

Este caso de uso tendrá lugar en un contexto en el que se llevarán a cabo multitud de accesos concurrentes que requerirán consultas de lectura sobre la base de datos. Un tiempo de respuesta que no exceda el tiempo de respuesta medio esperado para una aplicación de estas características es esencial para no degradar la experiencia de usuario (Figura 3.4)(Figura 3.5).

Usabilidad

Incrementar la facilidad a la hora de realizar la acción de filtrado es necesario para que el usuario se sienta satisfecho y confiado cuando utiliza nuestro sistema. Esto pretende evitar cualquier tipo de abandono debido a dificultades que no tengan que ver con el contenido del curso, las cuales son las que a nosotros nos atañen (Figura 3.6)(Figura 3.7).

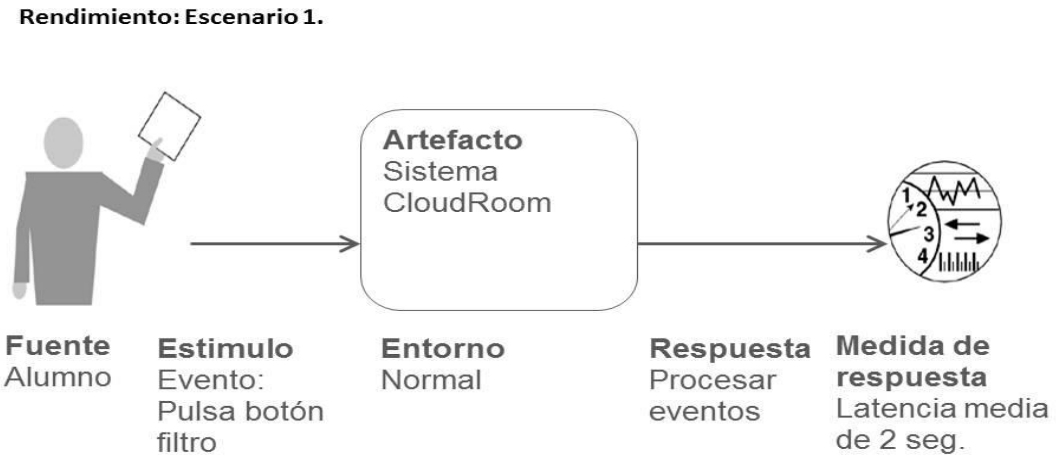


Figura 3.4 *Caso Uso 2: Escenario de Rendimiento 1.*

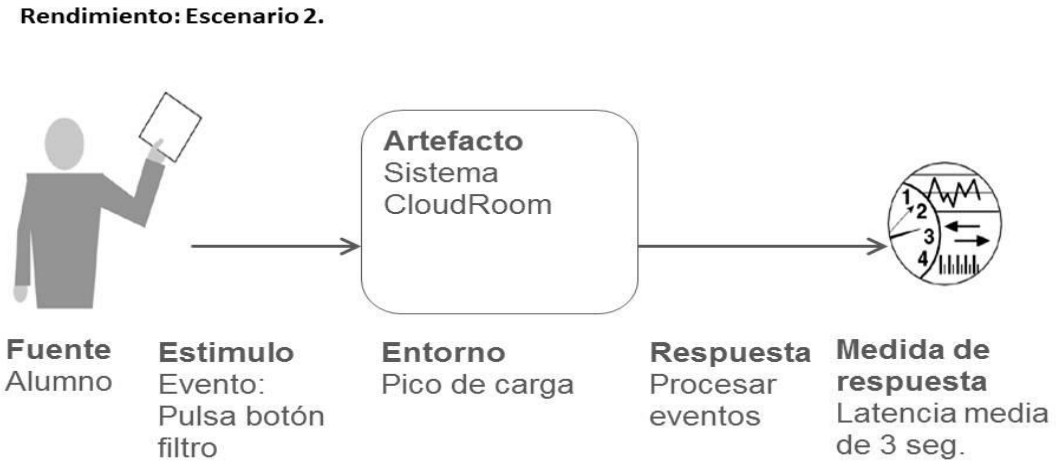


Figura 3.5 *Caso Uso 2: Escenario de Rendimiento 2.*

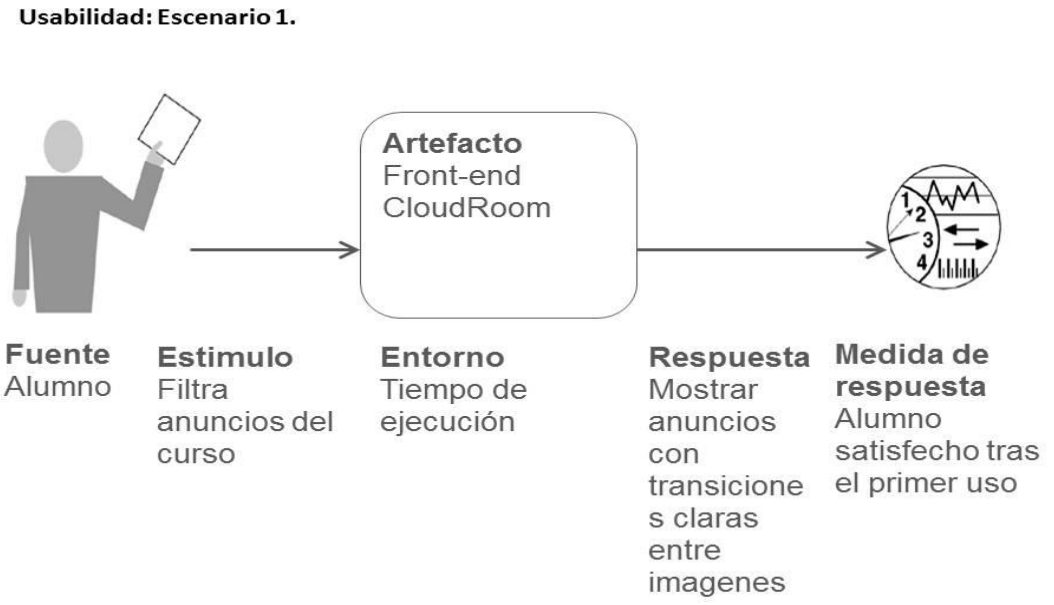


Figura 3.6 *Caso Uso 2: Escenario de Usabilidad 1.*

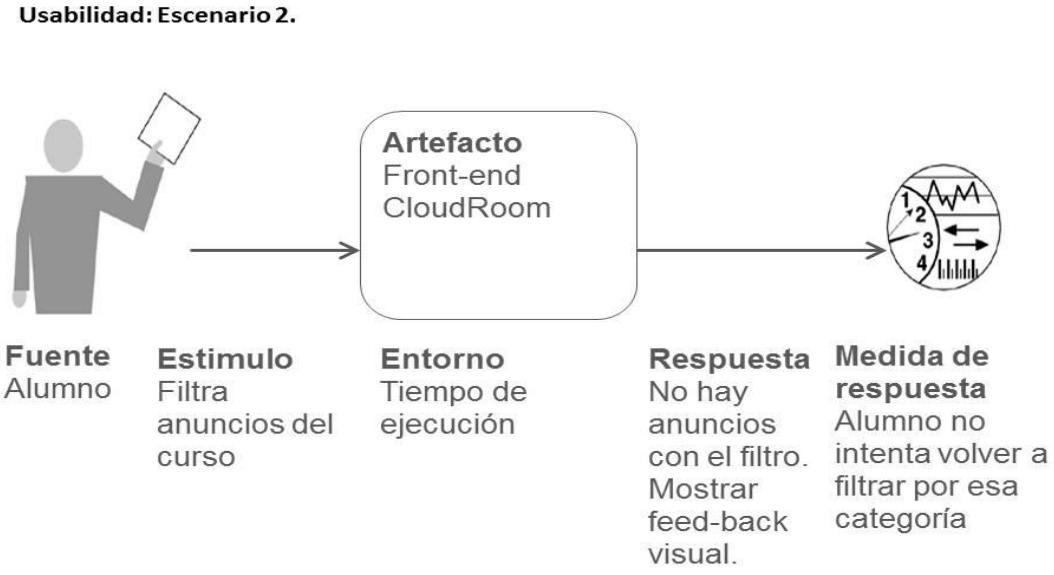


Figura 3.7 *Caso Uso 2: Escenario de Usabilidad 2.*

Caso de Uso 3: Navegar por el contenido del curso.

El alumno podrá desplegar un menú de navegación que le permita navegar por las lecciones que componen el curso y en cada una de estas lecciones también será capaz de navegar por el contenido de cada una de ellas. Esto le permitirá tener una perspectiva global de todo el curso y visitar los contenidos que desee en cualquier momento (Tabla 3.4).

Caso de Uso ID:	3		
Caso de Uso Nombre:	Navegar por el contenido del curso		
Creado por:	Carlos García	Actualizado por:	
Fecha creación:	14/04/2014	Fecha actualización:	
Descripción:	El alumno entra en el curso en el cual se ha registrado previamente y accede a la lección que desea seguir. Una vez en la lección puede navegar por el contenido (videos, tests o apuntes) de manera sencilla y eficiente.		
Actores:	Alumno		
Disparador:	El alumno accede al contenido del curso.		
Precondiciones:	1. El alumno ha accedido a un curso en el que está registrado. 2. El alumno debe estar logueado en el sistema.		
Postcondiciones:	1. El sistema recordará en que contenido se encuentra el alumno.		
Escenario normal:	3.0.1 El alumno accede al curso en el que está registrado. 3.0.2 El alumno accede al contenido del curso. 3.0.3 El sistema muestra un menú con el desglose de las lecciones y el contenido de la lección donde se quedó la última vez. 3.0.4 El alumno selecciona el siguiente contenido dentro de la lección. 3.0.5 El sistema muestra el nuevo contenido. 3.0.6 El alumno selecciona otra lección. 3.0.7 El sistema muestra el primer contenido de la lección seleccionada. 3.0.8 El alumno sale del curso.		

Escenarios alternativos:	Escenario alternativo 1 El alumno accede al contenido desordenadamente y el sistema lo muestra sin ningún problema.
Excepciones:	3.0.E.1 El alumno no está registrado en el curso. No podrá acceder a este y se le comunicará con un mensaje en pantalla. 3.0.E.2 El sistema no es capaz de cargar el contenido. Se le notificará al alumno con un mensaje por pantalla.
Includes:	
Prioridad:	Alta: Poder navegar por el contenido del curso libremente y de forma sencilla mejora la experiencia de usuario.
Frecuencia de uso:	1 vez cada 3 min por sesión
Complejidad:	Media

Tabla 3.4 *Caso de Uso 3: Navegar por el contenido del curso.*

Requisitos funcionales.

- Como alumno quiero ver un despliegue de las lecciones del curso.
- Como alumno quiero ver un despliegue del contenido de cada lección.
- Como alumno quiero saber de qué trata cada lección y contenido sin tener que acceder a este.
- Como alumno quiero navegar por el contenido de la lección.
- Como alumno quiero navegar por las lecciones del curso.

Requisitos no-funcionales.

Rendimiento.

Una navegación fluida aumenta la experiencia de usuario evitando distracciones innecesarias a la hora de seguir el contenido del curso (Figura 3.8)(Figura 3.9).

Usabilidad.

Proveer de un buen *feed-back* al alumno es muy importante para que la navegación sea intuitiva y así evitar perderse en esta (Figura 3.10)(Figura 3.11).

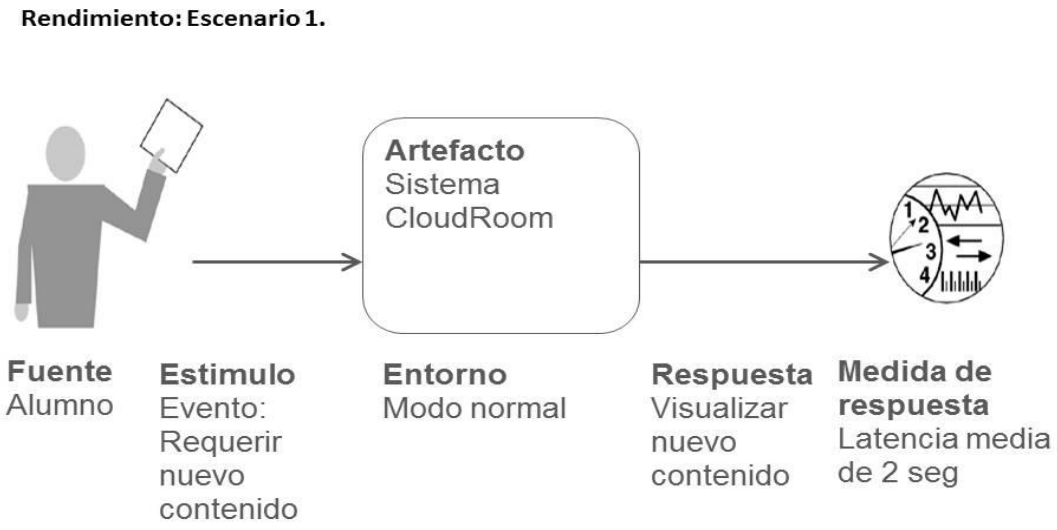


Figura 3.8 *Caso Uso 3: Escenario de Rendimiento 1.*

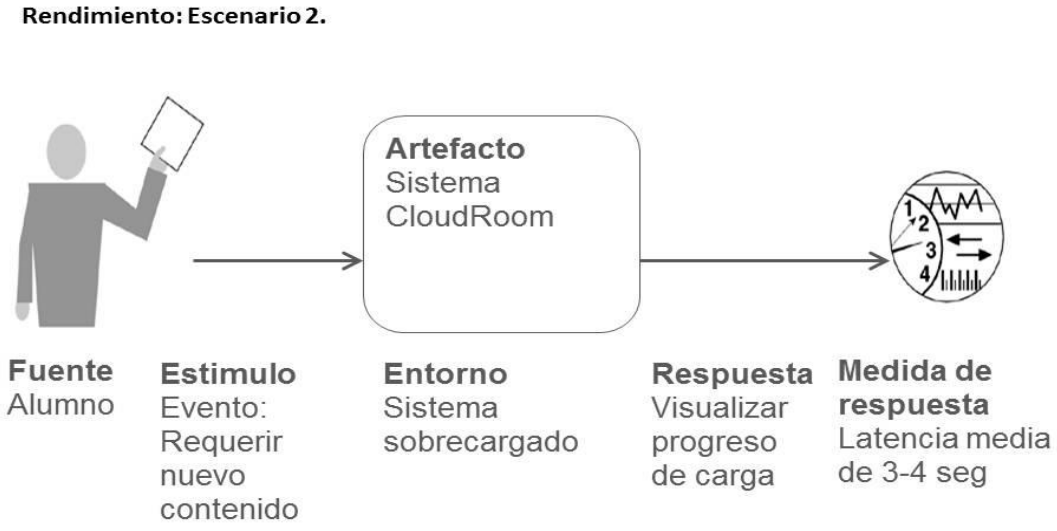


Figura 3.9 *Caso Uso 3: Escenario de Rendimiento 2.*

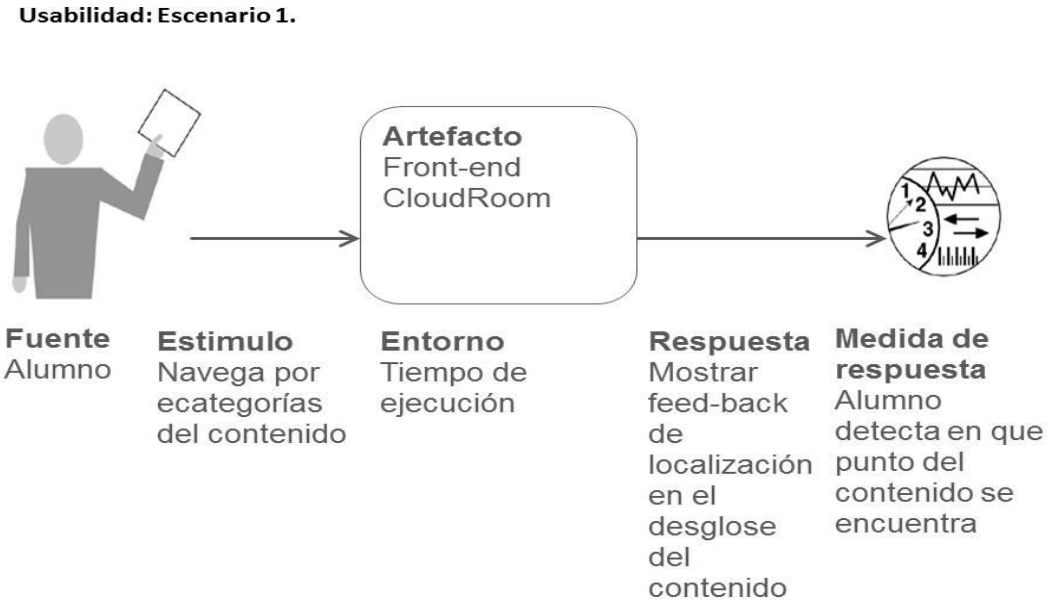


Figura 3.10 *Caso Uso 3: Escenario de Usabilidad 1.*

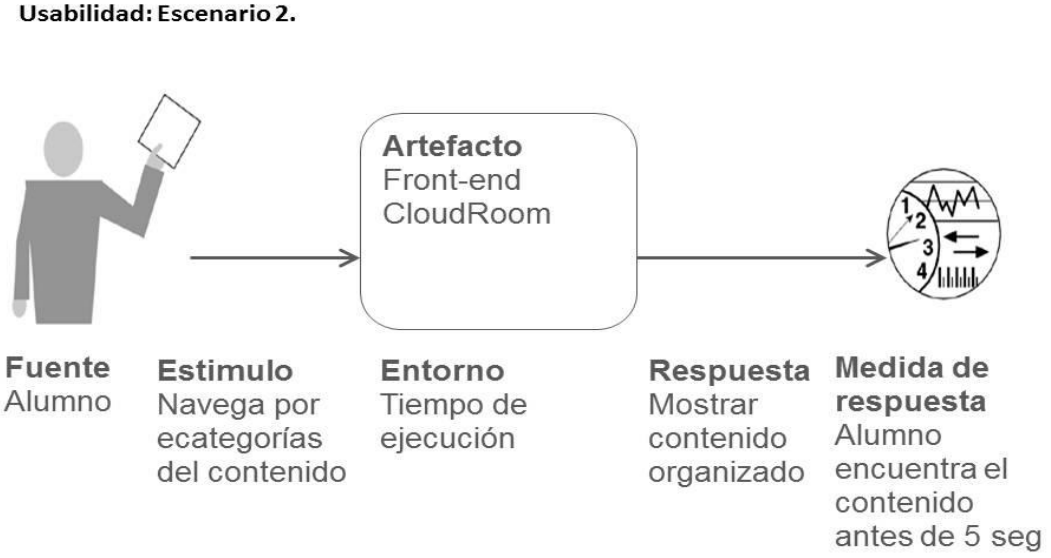


Figura 3.11 *Caso Uso 3: Escenario de Usabilidad 2.*

Caso de Uso 4: Ver vídeo de la lección.

El alumno podrá visualizar videos como parte del contenido del curso. Ver videos permite al alumno seguir el curso fácilmente y mantiene la atención del alumno. La plataforma de visualización de videos será interactiva y permitirá al alumno contestar a cuestiones embebidas dentro del vídeo (Tabla 3.5).

Caso de Uso ID:	4		
Caso de Uso Nombre:	Ver video de la lección		
Creado por:	Carlos García	Actualizado por:	
Fecha creación:	14/04/2014	Fecha actualización:	

Descripción:	El alumno accederá al contenido de la lección, el cual se tratará de un vídeo interactivo capaz de plantear cuestiones que el alumno deberá resolver en mitad del video.
Actores:	Alumno
Disparador:	El alumno reproduce un video.
Precondiciones:	1. El alumno debe estar logueado en el sistema. 2. El alumno debe haber accedido a un curso en el que esté registrado.
Postcondiciones:	1. El sistema marcará el contenido como completado.
Escenario normal:	4.0.1 El alumno accede al curso en el que está registrado. 4.0.2 El alumno accede al contenido del curso. 4.0.3 El sistema muestra un menú con el desglose de las lecciones y el contenido de la lección donde se quedó la última vez. 4.0.4 El alumno inicia el vídeo que hay como contenido. 4.0.5 El sistema reproduce el video. 4.0.6 El video termina. 4.0.7 El alumno sale del curso.
Escenarios alternativos:	Escenario alternativo 1 El alumno para el video y sale de ese contenido. Más tarde el alumno vuelve a ese contenido y continua reproduciendo el vídeo por donde se quedó.

Excepciones:	4.0.E.1 El alumno no está registrado en el curso. No podrá acceder a este y se le comunicará con un mensaje en pantalla. 4.0.E.2 El sistema no es capaz de cargar el video. Se le notificará al alumno con un mensaje por pantalla.
Includes:	
Prioridad:	Alta: Poder visualizar videos es imprescindible en los MOOCs en la actualidad.
Frecuencia de uso:	5-6 veces por sesión
Complejidad:	Alta

Tabla 3.5 *Caso de Uso 4: Ver vídeo de la lección.*

Requisitos funcionales.

- Como alumno quiero ver vídeos explicativos de la lección.

Requisitos no-funcionales.

Disponibilidad.

Proporcionar una calidad de vídeo buena y durante el mayor tiempo posible, evitando largas esperas o perdida de contenido visual es esencial a la hora de proveer a los cursos con vídeos. En este caso, la disponibilidad del contenido multimedia no afecta al diseño del módulo de cursos, ya que según el SLA (*Service Level Agreement*), se encarga *Amazon Web Services S3* donde se encuentra el *back-end* de nuestra aplicación.

Rendimiento.

Cortos tiempos de espera son esenciales hoy en día en aplicaciones distribuidas. Un largo tiempo de espera hace que el usuario abandone tu aplicación (Figura 3.12)(Figura 3.13).

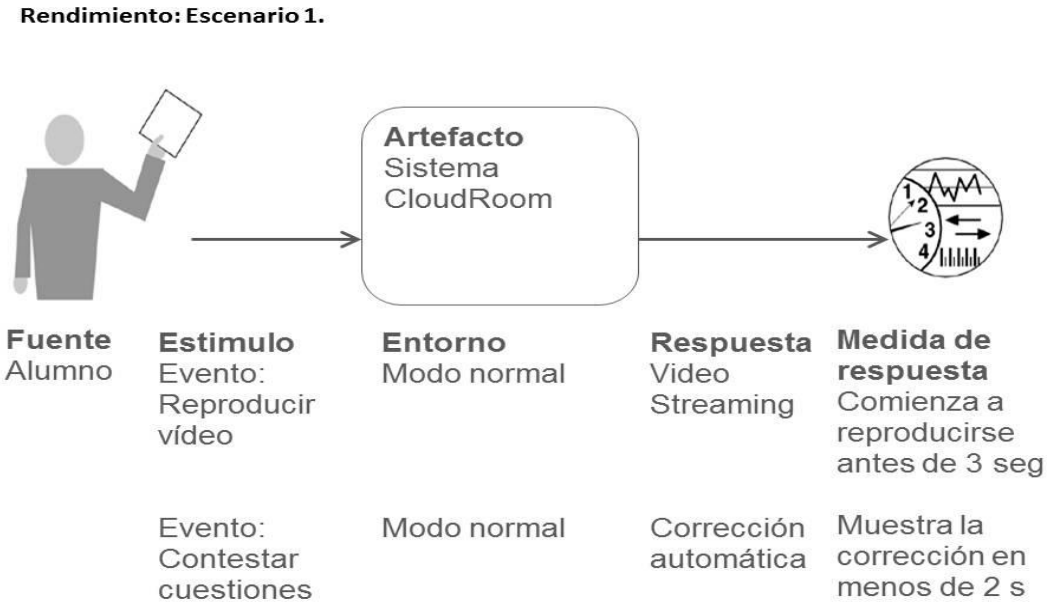


Figura 3.12 *Caso Uso 4: Escenario de Rendimiento 1.*

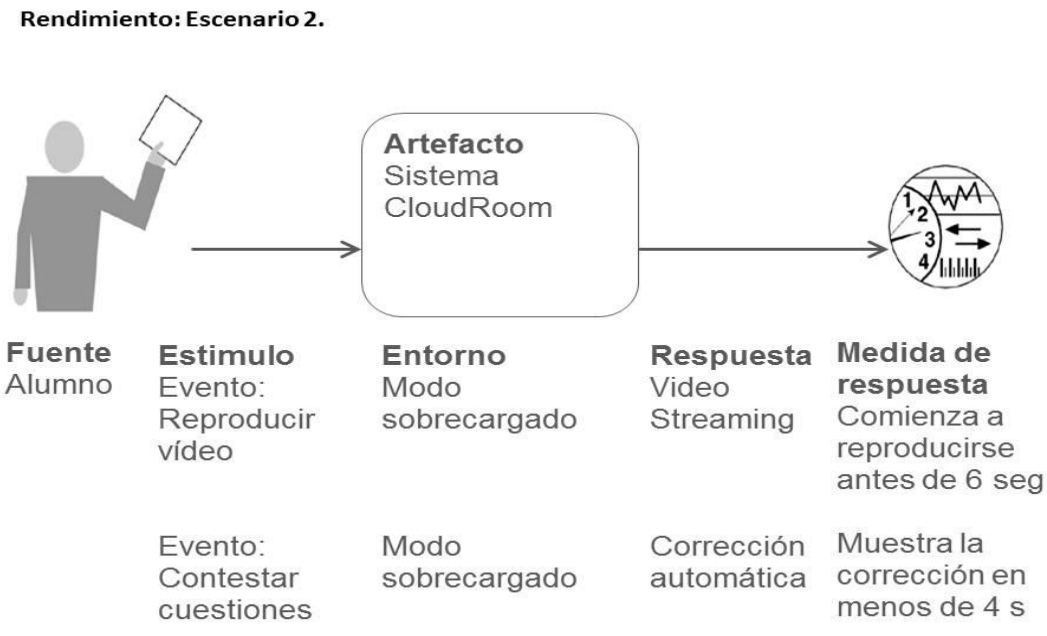


Figura 3.13 *Caso Uso 4: Escenario de Rendimiento 2.*

Caso de Uso 5: Ver apuntes de la lección.

El alumno podrá visualizar apuntes como parte del contenido del curso. Los apuntes servirán para facilitar el seguimiento del curso a los alumnos que no se sientan a gusto con el aprendizaje mediante vídeos. Además, podrá servir también como complemento a los vídeos para todos aquellos alumnos que si deseen seguir los vídeos (Tabla 3.6).

Caso de Uso ID:	5		
Caso de Uso Nombre:	Ver apuntes de la lección		
Creado por:	Carlos García	Actualizado por:	
Fecha creación:	14/04/2014	Fecha actualización:	

Descripción:	El alumno accederá al contenido de la lección, el cual se tratará de unos apuntes o de un vídeo interactivo más apuntes. Y los visualizará como parte del contenido de la lección.
Actores:	Alumno
Disparador:	Al navegar por el contenido el alumno accede a un contenido con apuntes.
Precondiciones:	1. El alumno debe estar logueado en el sistema. 2. El alumno debe haber accedido a un curso en el que esté registrado.
Postcondiciones:	1. El sistema marcará el contenido como visitado a modo de indicar el progreso que lleva el alumno.
Escenario normal:	5.0.1 El alumno accede al curso en el que está registrado. 5.0.2 El alumno accede al contenido del curso. 5.0.3 El sistema muestra un menú con el desglose de las lecciones y el contenido de la lección donde se quedó la última vez. 5.0.4 El alumno visualiza los apuntes que hay como contenido. 5.0.5 El alumno continúa por el siguiente contenido de la lección. 5.0.6 El alumno sale del curso.
Escenarios alternativos:	

Excepciones:	5.0.E.1 El alumno no está registrado en el curso. No podrá acceder a este y se le comunicará con un mensaje en pantalla. 5.0.E.2 El sistema no es capaz de cargar los apuntes. Se le notificará al alumno con un mensaje por pantalla.
Includes:	Caso de Uso 6: Realizar test del contenido.
Prioridad:	Alta: Poder visualizar apuntes es imprescindible en los MOOCs en la actualidad.
Frecuencia de uso:	5-6 veces por sesión
Complejidad:	Baja

Tabla 3.6 *Caso de Uso 5: Ver apuntes de la lección.*

Requisitos funcionales.

- Como alumno quiero ver el contenido de una lección con apuntes.
- Como alumno quiero ver el contenido de una lección con apuntes que complementen a un vídeo.

Requisitos no-funcionales.

Disponibilidad.

Proporcionar contenido multimedia y durante el mayor tiempo posible, evitando largas esperas o perdida de contenido visual es esencial a la hora de proveer a los cursos con vídeos. En este caso, la disponibilidad del contenido multimedia no afecta al diseño del módulo de cursos, ya que según el SLA (*Service Level Agreement*), se encarga *Amazon Web Services S3* donde se encuentra el *back-end* de nuestra aplicación.

Rendimiento.

Cortos tiempos de espera son esenciales hoy en día en aplicaciones distribuidas. Los largos tiempos de espera hacen que la interacción entre el usuario y la aplicación sea poco fluida provocando una mala experiencia de usuario, la cual es esencial en una plataforma educativa, ya que se pretende que el usuario se sienta cómodo para evitar abandonos (principal problema en estas plataformas) (Figura 3.14)(Figura 3.15).

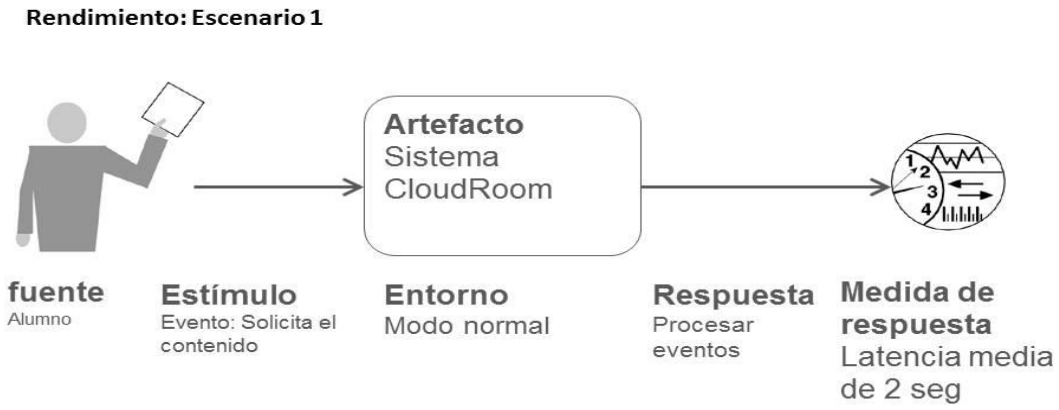


Figura 3.14 *Caso Uso 5: Escenario de Rendimiento 1.*

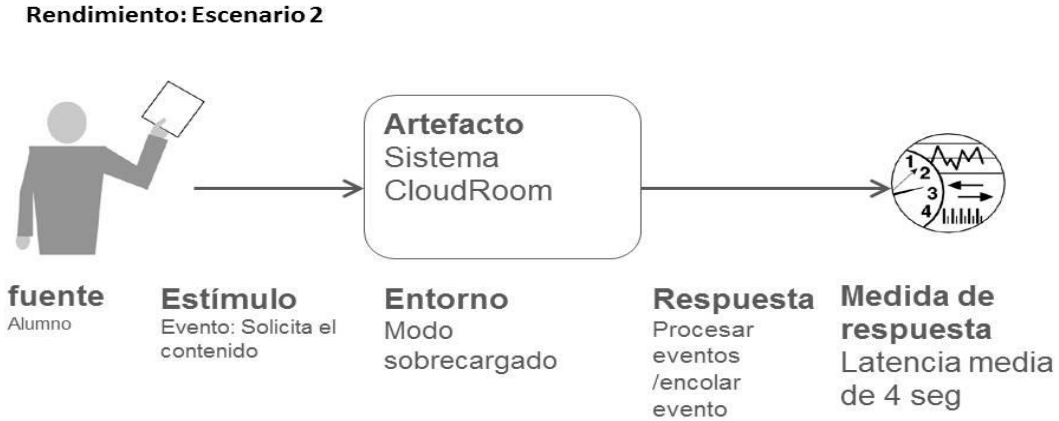


Figura 3.15 *Caso Uso 5: Escenario de Rendimiento 2.*

Caso de Uso 6: Realizar test del contenido.

Realizar test de seguimiento del contenido visto hasta el momento facilita el seguimiento del curso, ya que provee a los alumnos de un *feedback* que les permite orientarse en cuanto al grado de conocimientos que han adquirido. Para ello, los alumnos deben poder conocer la corrección a instante, y poder solicitar las respuestas correctas (Tabla 3.7).

Caso de Uso ID:	6		
Caso de Uso Nombre:	Realizar test del contenido.		
Creado por:	Carlos García	Actualizado por:	
Fecha creación:	14/04/2014	Fecha actualización:	
Descripción:	El alumno irá recorriendo los contenidos de una lección hasta que uno de ellos se trate de un test. Este test evaluará los contenidos previos vistos hasta ese momento. El alumno realizará el test y podrá conocer la corrección justo al terminarlo. También tendrá la opción de conocer las respuestas correctas.		
Actores:	Alumno		
Disparador:	Al navegar por el contenido el alumno accede a un contenido que se trata de un test.		
Precondiciones:	1. El alumno debe estar logueado en el sistema. 2. El alumno debe haber accedido a un curso en el que esté registrado.		
Postcondiciones:	1. El sistema marcará el contenido como visitado a modo de indicar el progreso que lleva el alumno.		
Escenario normal:	6.0.1 El alumno accede al curso en el que está registrado. 6.0.2 El alumno accede al contenido del curso. 6.0.3 El sistema muestra un menú con el desglose de las lecciones y el contenido de la lección donde se quedó la última vez. 6.0.4 El selecciona como contenido un test. 6.0.5 El sistema muestra el test. 6.0.6 El alumno responde a las preguntas del test. 6.0.7 El alumno pulsa el botón para enviar el test respondido.		

	6.0.8 El sistema responde marcando que preguntas están bien y cuales mal. 6.0.9 El alumno continúa por el siguiente contenido de la lección. 6.0.10 El alumno sale del curso.
Escenarios alternativos:	Escenario alternativo 1 El alumno una vez conoce que respuestas tiene mal, desea conocer cuáles son las correctas, así que solicita las correctas y el sistema las muestra.
Excepciones:	6.0.E.1 El alumno no está registrado en el curso. No podrá acceder a este y se le comunicará con un mensaje en pantalla. 6.0.E.2 El sistema no es capaz de cargar el test. Se le notificará al alumno con un mensaje por pantalla.
Includes:	
Prioridad:	Alta: Poder realizar test de seguimiento de lo visto hasta ahora facilita el seguimiento y aprendizaje en el curso.
Frecuencia de uso:	2-3 veces por sesión
Complejidad:	Media

Tabla 3.7 *Caso de Uso 6: Realizar test del contenido.*

Requisitos funcionales.

- Como alumno quiero realizar test acerca del contenido visto.
- Como alumno quiero saber la corrección del test al finalizarlo.
- Como alumno quiero conocer las respuestas correctas.

Requisitos no-funcionales.

Rendimiento.

Buscamos tiempos de espera cortos cuando el alumno interactúa con la aplicación. La comprobación de las respuestas no debe llevar mucho tiempo. Tener que almacenar las respuestas en la base de datos puede suponer un incremento en la latencia no deseado, por tanto reducir esa sobrecarga es necesario para incrementar la experiencia de usuario (Figura 3.16)(Figura 3.17).

Usabilidad.

La metodología de respuesta y comprobación en los test debe ser sencilla para evitar que el alumno pierda tiempo aprendiendo a hacerlo y hacer así más eficiente su interacción con la aplicación (Figura 3.18)(Figura 3.19).

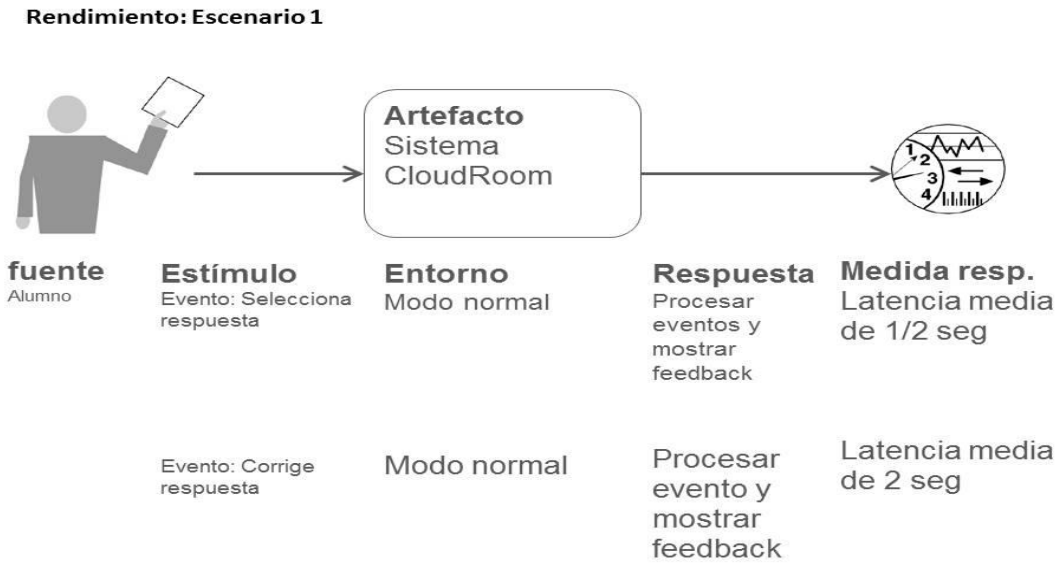


Figura 3.16 *Caso Uso 6: Escenario de Rendimiento 1.*

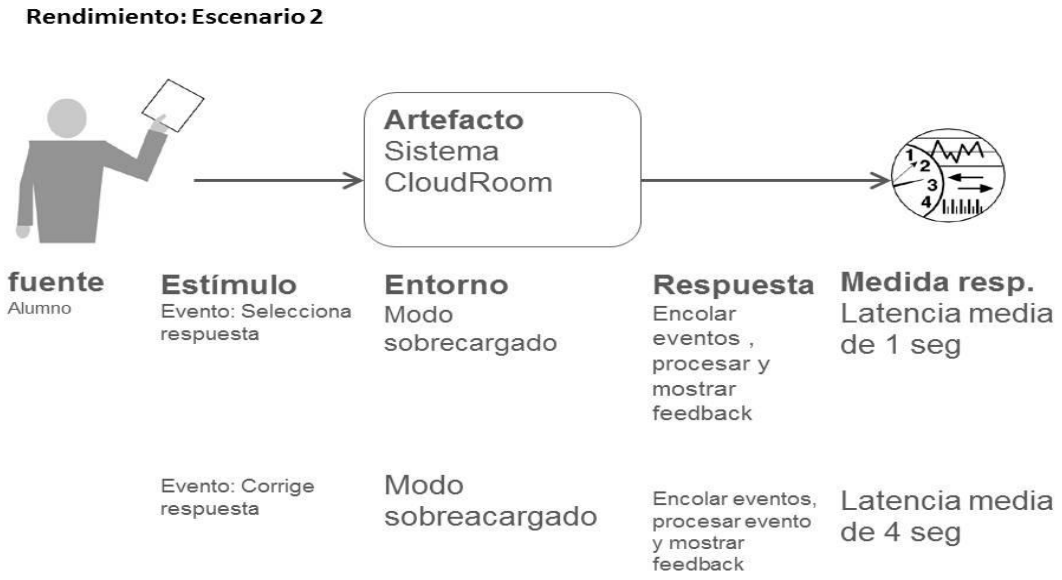


Figura 3.17 *Caso Uso 6: Escenario de Rendimiento 2.*

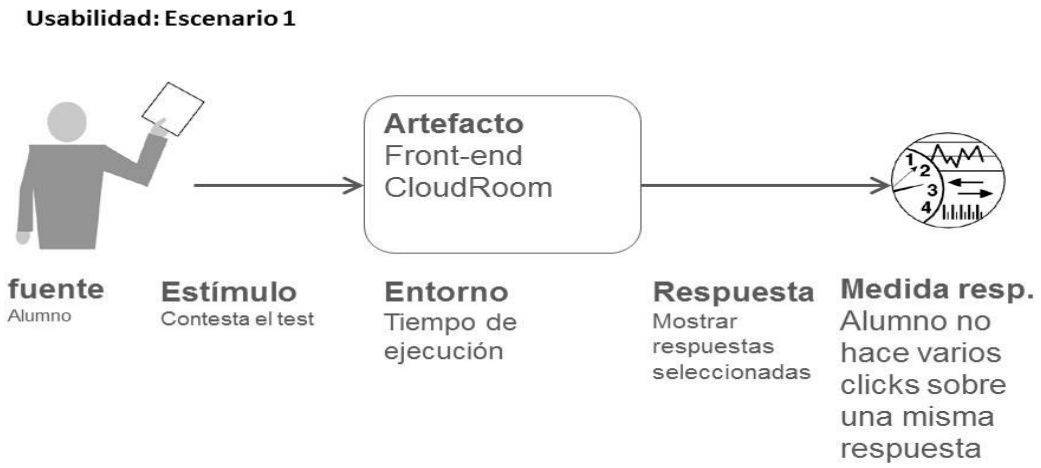


Figura 3.18 *Caso Uso 6: Escenario de Usabilidad 1.*

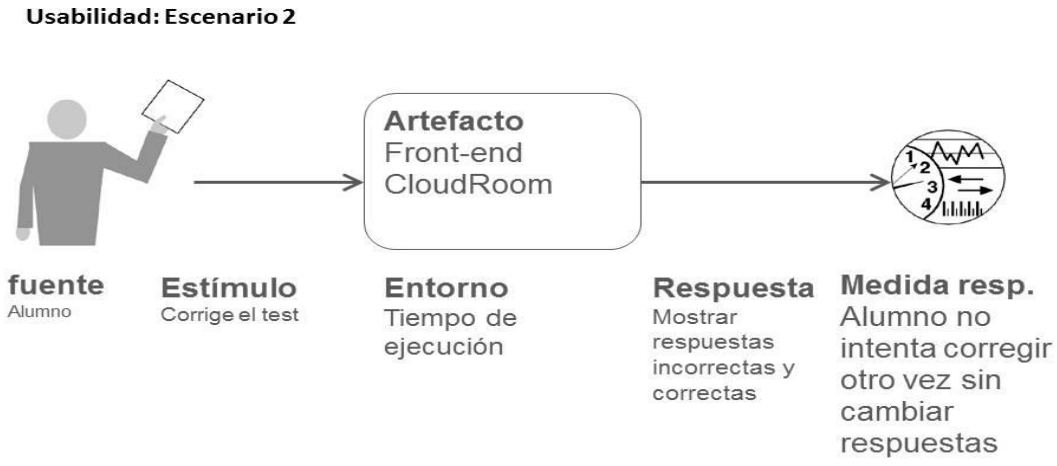


Figura 3.19 *Caso Uso 6: Escenario de Usabilidad 2.*

Caso de Uso 7: Realizar quiz.

Se trata de responder a una cuestión embebida en un vídeo. De esta forma se mantiene el interés del alumno durante el vídeo y le permite evaluar los conceptos vistos en el vídeo. La cuestión al aparecer para el vídeo y tiene que ser contestada por el alumno sin límite de tiempo. Una vez contestada puede comprobar el resultado y de ser incorrecto saltar al momento del vídeo donde el profesor explica la respuesta correcta (Tabla 3.8).

Caso de Uso ID:	7		
Caso de Uso Nombre:	Realizar quiz.		
Creado por:	Carlos García	Actualizado por:	
Fecha creación:	14/04/2014	Fecha actualización:	

Descripción:	El alumno reproduce un vídeo de un capítulo donde se explican nuevos conceptos. En un momento determinado el vídeo se para y aparece una cuestión sobre este que debe ser contestada de forma interactiva por el alumno. Una vez contestada, el alumno puede saltar a la parte del vídeo que explica la respuesta correcta o seguir por la parte del vídeo que explica nuevos conceptos.
Actores:	Alumno
Disparador:	El alumno llega a un punto de un vídeo donde aparece una cuestión.
Precondiciones:	1 El alumno debe estar logueado en el sistema. 2 El alumno debe haber accedido a un curso en el que esté registrado.
Postcondiciones:	1. El sistema marcará el quiz como realizado a modo de indicar el progreso que lleva el alumno.
Escenario normal:	7.0.1 El alumno accede al curso en el que está registrado. 7.0.2 El alumno accede al contenido del curso. 7.0.3 El sistema muestra un menú con el desglose de las lecciones y el contenido de la lección donde se quedó la última vez. 7.0.4 El alumno selecciona como contenido un vídeo. 7.0.5 El sistema reproduce el vídeo. 7.0.6 El vídeo se para y aparece un quiz.

	7.0.7 El alumno responde el quiz. 7.0.8 El alumno pulsa el botón para comprobar las respuestas. 7.0.9 El sistema muestra si la respuesta es correcta o no. 7.0.10 El contenido queda marcado como visitado. 7.0.11 El alumno pulsa el botón para acceder al punto del vídeo donde se explica la respuesta correcta. 7.0.12 El alumno termina de ver el vídeo. 7.0.13 El alumno abandona el sistema.
Escenarios alternativos:	Escenario alternativo 1 El alumno una vez conoce que respuestas tiene mal, en lugar de pasar a la parte del vídeo donde se explica la respuesta, salta a la parte donde continúan explicándose nuevos coceptos.
Excepciones:	7.0.E.1 El alumno no está registrado en el curso. No podrá acceder a este y se le comunicará con un mensaje en pantalla. 7.0.E.2 El sistema no es capaz de cargar el test. Se le notificará al alumno con un mensaje por pantalla. 7.0.E.3 El alumno avanza el vídeo saltando el quiz. Se le notificará con un mensaje.
Includes:	
Prioridad:	Alta: Poder realizar quiz de seguimiento de lo visto hasta ahora facilita el seguimiento y aprendizaje en el curso.
Frecuencia de uso:	2-3 veces por sesión
Complejidad:	Media

Tabla 3.8 *Caso de Uso 7: Realizar quiz.*

Requisitos funcionales.

- Como alumno quiero realizar quiz embebidos en el vídeo.
- Como alumno quiero conocer las respuestas correctas.
- Como alumno quiero continuar el vídeo sin conocer la explicación del quiz.
- Como alumno quiero conocer la explicación de la respuesta correcta de un quiz.

Requisitos no-funcionales.

Rendimiento.

Del mismo modo que con los tests, precisamos de tiempos de espera cortos cuando el alumno interactúa con la aplicación. Es decir, comprobaciones de respuesta rápida y

almacenamiento de lo respondido en la base de datos sin mucha sobrecarga en el rendimiento (Figura 3.20)(Figura 3.21).

Usabilidad.

La metodología de respuesta y comprobación en los test debe ser sencilla para evitar que el alumno pierda tiempo aprendiendo a hacerlo y hacer así más eficiente su interacción con la aplicación. Agilizar el seguimiento del curso, evitando que el alumno se vea obligado a ver una explicación que ya ha comprendido, o dejar sin explicación la pregunta puede dañar la experiencia de usuario (Figura 3.22)(Figura 3.23).

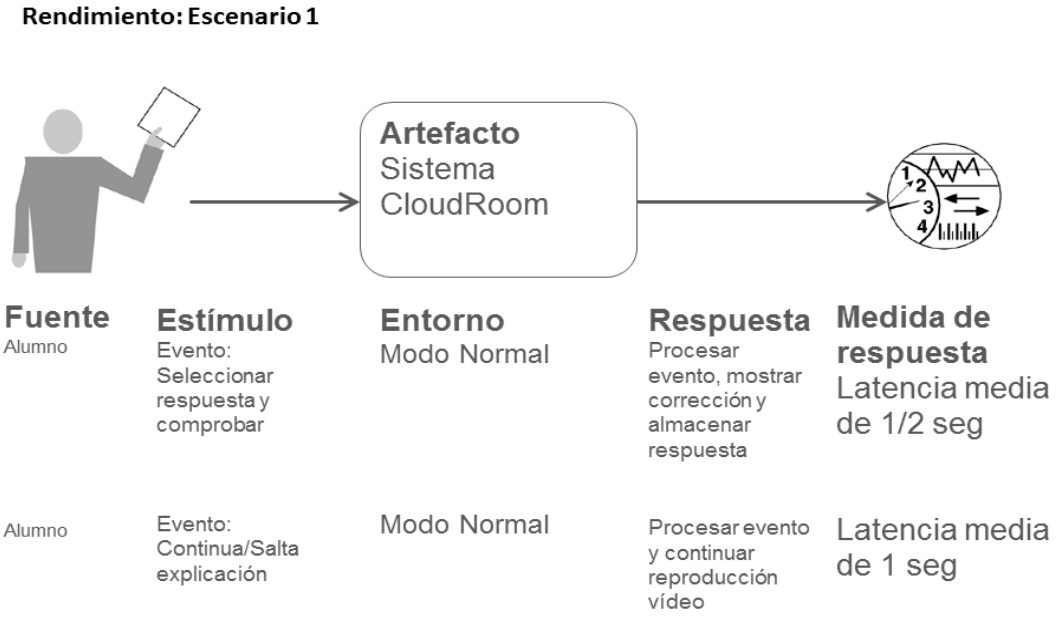


Figura 3.20 *Caso Uso 7: Escenario de Rendimiento 1.*

Figura 3.21 *Caso Uso 7: Escenario de Rendimiento 2.*

Figura 3.22 *Caso Uso 7: Escenario de Usabilidad 1.*

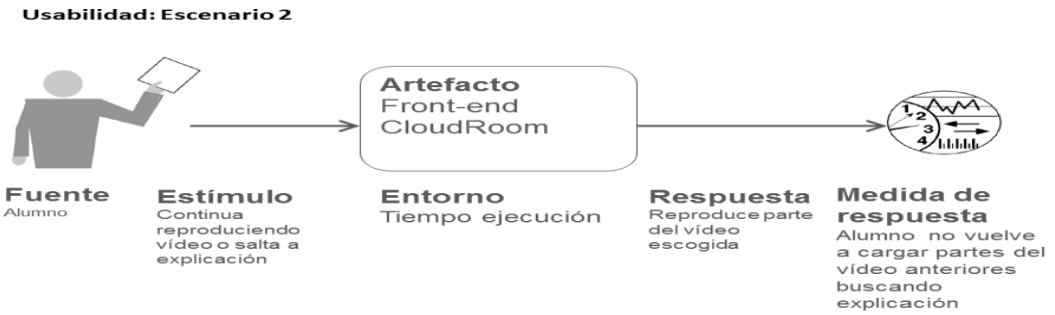


Figura 3.23 *Caso Uso 7: Escenario de Usabilidad 2.*

3.2 Requisitos funcionales: Historias de Usuario.

Una vez definidos los Casos de Uso, procedimos a recolectar todas las historias de usuario que fueron surgiendo.

En primer lugar, se creó una trazabilidad entre historias de usuario y casos de uso para que cualquiera encargado de desarrollar la historia de usuario pudiera consultar la documentación referente al requisito si fuera necesario. Por tanto, cada historia de usuario fue anotada con un prefijo" CLOUD _CURSOS_x.CUy", donde la 'x' es el número de la historia de usuario dentro del desarrollo del módulo de cursos y la 'y' es el número del Caso de Uso (Tabla 3.9).

Por último, a cada historia de usuario se le asignó una prioridad en función del valor que tenía ésta para el usuario y se introdujo en el Product Backlog (pizarra con todas las historias de usuario de la aplicación priorizadas) con el resto de historias de usuario.

Etiqueta	Historia de Usuario	Prioridad
CLOUD_CURSOS_1.CU1	Como alumno quiero ver un listado de todos los anuncios relativos al curso ordenados cronológicamente empezando por los más actuales	ALTA
CLOUD_CURSOS_2.CU1	Como alumno quiero obtener los anuncios más antiguos además de los que ya estoy viendo.	ALTA
CLOUD_CURSOS_3.CU2	Como alumno quiero filtrar los anuncios por anuncios de institutciones.	ALTA
CLOUD_CURSOS_4.CU2	Como alumno quiero filtrar los anuncios por ejercicios.	ALTA
CLOUD_CURSOS_5.CU2	Como alumno quiero filtrar los anuncios por anuncios del profesor	ALTA
CLOUD_CURSOS_6.CU3	Como alumno quiero ver un despliegue de las lecciones del curso	ALTA
CLOUD_CURSOS_7.CU3	Como alumno quiero ver un despliegue del contenido de cada lección.	ALTA
CLOUD_CURSOS_8.CU3	Como alumno quiero saber de que trata cada lección y contenido sin tener que acceder a este.	ALTA
CLOUD_CURSOS_9.CU3	Como alumno quiero navegar por el contenido de la lección.	ALTA
CLOUD_CURSOS_10.CU3	Como alumno quiero navegar por las lecciones del curso.	ALTA

Etiqueta	Historia de Usuario	Prioridad
CLOUD_CURSOS_11.CU4	Como alumno quiero ver vídeos explicativos de la lección.	ALTA
CLOUD_CURSOS_12.CU5	Como alumno quiero ver el contenido de una lección con apuntes enriquecidos.	ALTA
CLOUD_CURSOS_13.CU5	Como alumno quiero ver el contenido de una lección con apuntes que complementen a los vídeos.	ALTA
CLOUD_CURSOS_14.CU6	Como alumno quiero realizar tests acerca del contenido del curso	ALTA
CLOUD_CURSOS_15.CU6	Como alumno quiero saber la corrección del test al finalizarlo.	ALTA
CLOUD_CURSOS_16.CU6	Como alumno quiero conocer las respuestas correctas tras la corrección.	ALTA
CLOUD_CURSOS_17.CU7	Como alumno quiero realizar quiz embebidos en el los vídeos.	ALTA
CLOUD_CURSOS_18.CU7	Como alumno quiero conocer las respuestas correctas tras la corrección.	ALTA
CLOUD_CURSOS_19.CU7	Como alumno quiero continuar el vídeo sin conocer la explicación del quiz.	ALTA
CLOUD_CURSOS_20.CU7	Como alumno quiero conocer la explicación de la respuesta correcta de un quiz.	ALTA

Tabla 3.9 *Listado de Historias de Usuario.*

3.3 Requisitos no-funcionales: ASRs.

A continuación hemos recogido una serie de atributos de calidad que presentan un fuerte impacto sobre la arquitectura denominados comúnmente *Architecturally Significant Requirements* (ASR) [19].

Cuando se construyó la arquitectura inicial se tuvo en consideración atributos de calidad como: disponibilidad, rendimiento, usabilidad y seguridad. Pero en lo que se refiere al módulo de cursos nos centraremos en el rendimiento, usabilidad y extensibilidad. Estos requisitos han surgido del análisis de requisitos realizado y de decisiones tomadas en cuanto a la posibilidad de introducir nuevos requisitos en el futuro.

Todo esto, se recoge en un *Utility Tree* (Tabla 3.23.10) que presenta los atributos de calidad más importantes y los ASRs, asignándoles una dupla (*valor para el usuario*, *impacto en la arquitectura*).

Atributo	**Objetivo**	**ASR**
Rendimiento	Latencia	El usuario recibirá un *feedback* durante la navegación en un periodo de 0.1 a 1 seg para tener una sensación de respuesta instantánea (Alta, Media). Con la aplicación se encuentre en un instante de sobrecarga, el usuario recibirá un *feedback* que le advierta de que su acción se está ejecutando correctamente y el sistema deberá responder en menos de 4 seg (Alta, Media).
Usabilidad	Operaciones normales	El usuario es capaz de seguir, orientarse y navegar por los contenidos de forma ágil tras la primera sesión (Media,Media).
	Usuario experimentado	Tras la primera semana el usuario no encuentra ninguna dificultad a la hora de interactuar con las funcionalidades del curso (Media, Media).
Extensibilidad	Añadir nueva funcionalidad	Dos nuevas historias de usuario son introducidas invirtiendo 1 persona/sprint (Media, Media).

Tabla 3.20 *Utility Tree.*

ARQUITECTURA

“La música es el espacio entre las notas.”

Claude Debussy

4 Solución propuesta.

En esta sección hablaremos de las herramientas y metodología de desarrollo software utilizados, y de las decisiones de diseño tomadas en cuanto a la arquitectura del *back-end* del módulo de cursos y al modelado de datos.

4.1 Metodología de desarrollo.

Debido a la intención de dirigir el producto hacia consumidores como un SaaS, decidimos aprovechar las ventajas que ofrecen los *SaaS* y el *Cloud* Computing a la hora de ser flexibles en la integración de nuevas versiones de software, ya que los requisitos de la aplicación tenían el riesgo de estar sujetos a cambios. Por ello, se adoptó una metodología de desarrollo ágil, donde utilizamos un desarrollo iterativo e incremental para desarrollar funcionalidad. Todo esto englobado en un ciclo PLAN-DO-CHECK [19], es decir, cada iteración representaba la introducción de nuevos requisitos, el diseño y documentación necesarios siguiendo el principio YAGNI (*You Ain't Gonna Need It*), y una validación y verificación finales cuyo *feedback* servía como entrada para la siguiente iteración.

4.1.1 Scrum.

Scrum [20] fue la metodología escogida para llevar a cabo el desarrollo. Se organizó un equipo de 4 personas donde cada uno se convirtió en responsable de uno de los módulos de CloudRoom. Aun así, en ningún momento se dependía de la organización de uno de los responsables, ya que el equipo estaba auto-organizado y en comunicación continua.

Esta auto-organización se basó en el establecimiento de un *Product Backlog* generado a partir de la extracción de requisitos con todas las Historias de Usuario (requisitos a implementar descritos desde una perspectiva de valor para el usuario), en sprints de 3 semanas cada uno dedicados a la implementación de Historias de Usuario y en una serie de reuniones durante todo el periodo de desarrollo alrededor de la pizarra Scrum (Figura 4.1). Una de las reuniones se hacía al principio del sprint, donde se planificaba que Historias de Usuario se implementarían durante el sprint, asignando a estas una puntuación en función de lo que el equipo estimase que iban a tardar en implementarla, esta puntuación, además, sirvió como medida de velocidad del equipo y como *feedback*

para las estimaciones de los futuros sprints. Luego, cada día se hacía una reunión de apenas 10 minutos donde se comentaba el progreso del sprint y los obstáculos encontrados. Y por último, había una reunión al finalizar el sprint que proporcionaba *feedback* acerca de lo que se debía mejorar.

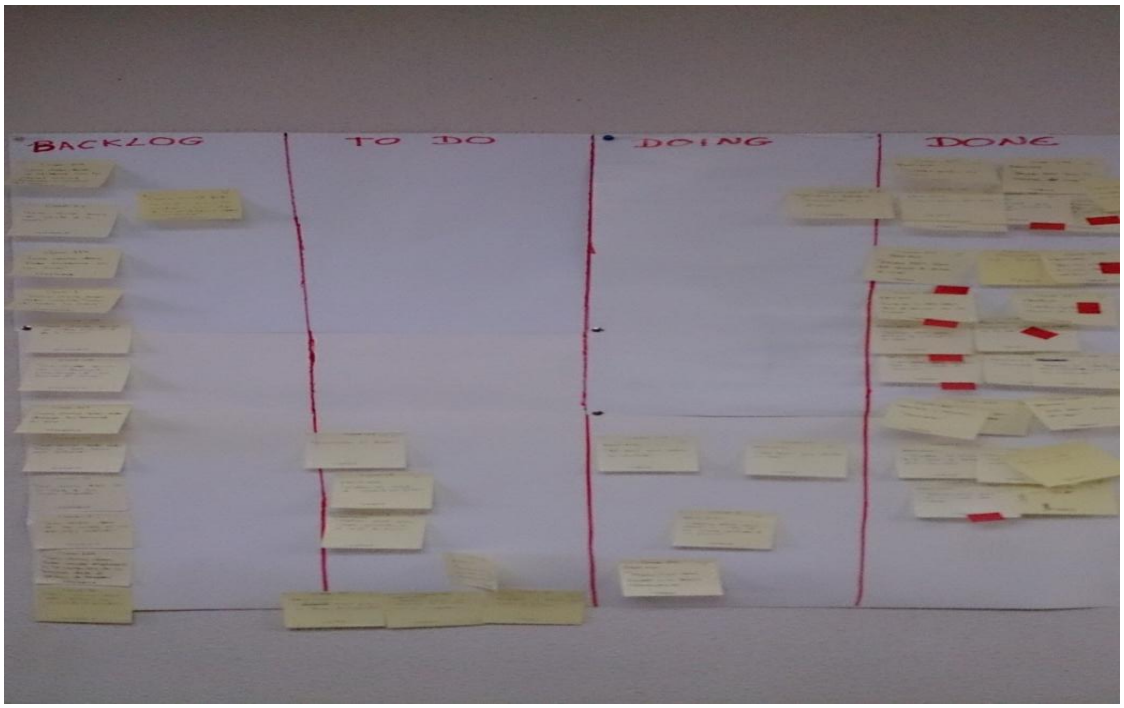


Figura 4.1 *Pizarra Scrum.*

4.1.2 Gestión de configuración y tareas.

Para la gestión de configuración utilizamos una serie de herramientas que automatizaron el control de versiones. Por un lado, empleamos Bitbucket como repositorio en la nube, por otro lado utilizamos la herramienta de control de versiones Git, por otro utilizamos Jira para la gestión de tareas y NPM (*Node Package Manager*) para la gestión de dependencias software.

Git

Cada desarrollador instaló la herramienta en su terminal de trabajo local y asoció su directorio de trabajo con la rama del repositorio de Bitbucket con la que iba a trabajar.

Jira

Esta herramienta automatizaba la asignación de tareas a los desarrolladores y la gestión del *Product Backlog* permitiendo trazabilidad con Bitbucket entre Historias de Usuario y ramas *Feature*.

Bitbucket

Con esta herramienta creamos dos repositorio de trabajo, uno para *front-end*, y otro para *back-end*. Cada uno de estos repositorios se organizó en las siguientes ramas de trabajo (Figura 4.2):

- ***Master***: en esta rama se alojaron las versiones que estarían en producción.
- ***Staging***: Esta rama es la más cercana a producción sin llegar a estar en producción y la utilizamos para pruebas de integración antes de pasar a producción.
- ***Develop***: En esta rama pusimos todas las Historias de Usuario una vez terminadas y que debían ser revisadas.
- ***Feature***: Se creaba una rama de este tipo por cada nueva Historia de Usuario que se fuese a implementar, y es en esta rama donde se desarrollaba.

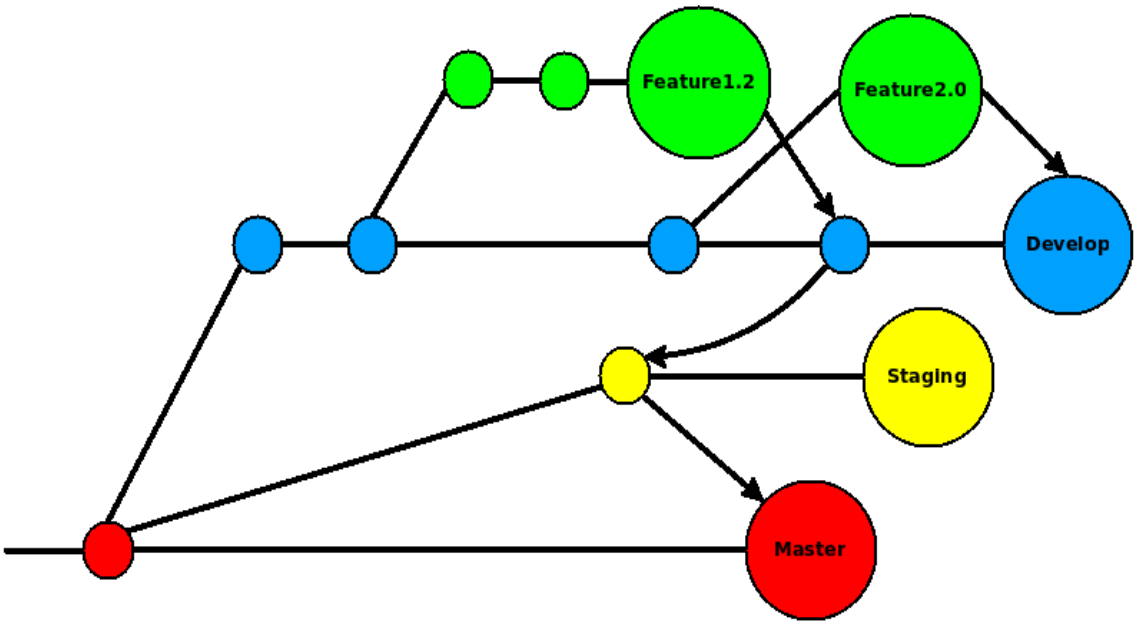


Figura 4.2 *Configuración ramas del repositorio de back-end.*

NPM (*Node Package Manager)*

Es el gestor de paquetes de Node.JS y facilita la instalación de dependencias definiendo el paquete y la versión de este en el fichero *package.json*, de modo que nos bastaba con ignorar el directorio con todas las dependencias (node_modules) en la herramienta de control de versiones Git a la hora de enviar código al repositorio Bitbucket. Después, simplemente habría que descargarse el código y ejecutar 'npm install' para instalar todas las dependencias descritas en *package.json*.

4.2 Diseño Software.

El proceso de diseño de los componentes software que constituyen la aplicación viene englobado en un proceso iterativo del tipo PLAN-DO-CHECK [19], donde la entrada que alimentará las decisiones de diseño viene de la etapa de análisis de requisitos. Teniendo esto en cuenta, los diseños serán validados contra los requisitos funcionales y no funcionales (atributos de calidad) y las restricciones impuestas por el tipo de arquitectura diseñada.

Por lo tanto, lo que hemos hecho ha sido instanciar componente a componente con un enfoque *top-down* donde empezamos por una visión global de la arquitectura por capas y nos vamos adentrando de forma más modular en cada capa.

Se ha partido de una arquitectura SOA donde se ha decidido que nuestro *back-end* sea un servicio RESTful ya que esto permite una gran escalabilidad e interoperabilidad entre nuestro *back-end* y la variedad de clientes en los que queremos permitir que se ejecute nuestra aplicación y los cuales estarán desarrollados en diferentes plataformas. Por tanto, el sistema se descompone en tres capas [21](Tabla 4.1):

Capa	**Responsabilidad**
Presentación	La componen los clientes que consumirán nuestro servicio (*back-end).*
Lógica del Dominio	Es el *back-end* de la aplicación donde está albergado nuestro servicio REST.
Persistencia	Esta capa la compone la base de datos de grafos NoSQL Neo4j I su interfaz REST es consumida por nuestro *back-end*.

Tabla 4.1 *Capas de la arquitectura del sistema.*

El alcance de este proyecto cubre el desarrollo de las dos últimas capas, Lógica del Dominio y Persistencia, por lo que a continuación pasaremos a explicar cómo se han diseñado los elementos que componen dichas capas.

4.2.1 Patrones arquitectónicos escogidos.

Para el diseño software, se escogieron en primera instancia los patrones arquitectónicos que mejor encajaban con las necesidades de la aplicación. Estos patrones son: *Service Layer* [21]que establece la arquitectura del *back-end* de forma monolítica, como sistema independiente con el que dialogaran los clientes, y *Data Mapper* [21]que establece la forma en la que se establecerá la comunicación dentro del *back-end* entre la capa de Lógica del Dominio y la capa de Persistencia.

Service Layer.

Establece las fronteras de nuestro *back-end* con una capa de servicios que envuelve toda la lógica de negocio. Implementa un conjunto de operaciones que actúan a modo de interfaz, y coordina la respuesta de la aplicación para cada operación. Dicha interfaz se implementará mediante una API REST.

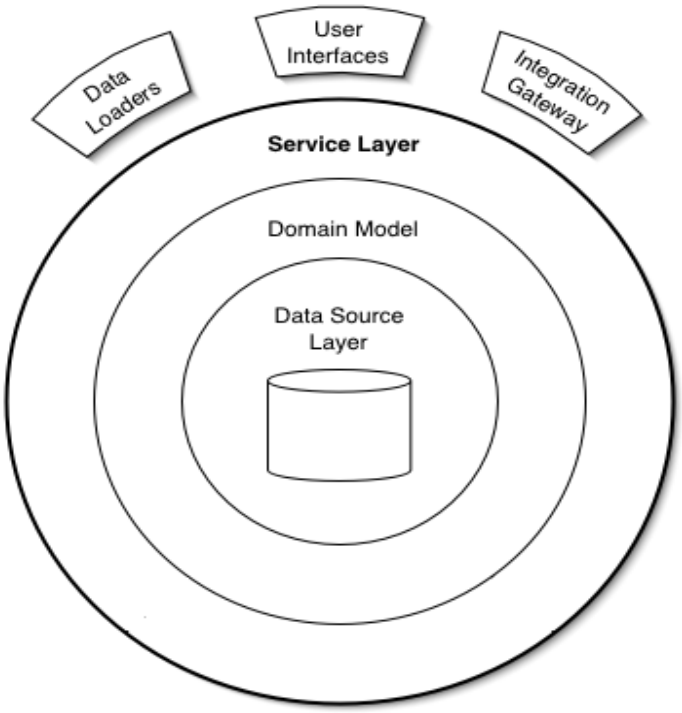


Figura 4.3 *Patrón arquitectónico Service Layer.*

La elección de este patrón viene promovida por la sobrecarga derivada de distribuir los componentes software. Las llamadas locales (mismo proceso o máquina) entre componentes son mucho más rápidas que las llamadas realizadas a diferentes procesos o

máquinas. Teniendo en cuenta la naturaleza de sistema distribuido de nuestra aplicación, es fácil observar que, aunque una interfaz más distribuida permite una gran flexibilidad, el tener que estar haciendo varias llamadas incrementa mucho la latencia, por tanto hemos creído necesario sacrificar esta flexibilidad en favor del rendimiento que ganaremos con una capa que envuelve las funcionalidades del *back-end* y las ofrece como una sola llamada, ya que estamos hablando de llamadas entre diferentes máquinas cuya latencia tiene un orden de magnitud mucho mayor que si fuera entre procesos.

Data Mapper

Debido a la naturaleza de *mashup* de nuestra aplicación, la cual tiene un gran número de usuarios potenciales, el primer atributo de calidad al que nos enfrentamos fue la escalabilidad. Por ello, se decidió implementar la capa de persistencia de nuestro sistema sobre una base de datos NoSQL de grafos, la cual favorece la escalabilidad de dicha capa facilitando la replicación y refactorización del modelo de datos.

Con esta restricción en la capa de persistencia nos vimos obligados a desechar algunos de los patrones arquitectónicos más comunes en las aplicaciones distribuidas en cuanto a la comunicación entre la capa de la lógica del dominio y la capa de persistencia. Los patrones desechados son aquellos que producían un mapeo directo con las entidades de la base de datos, como son: *Table Data Gateway* o *Active Record* que producen instancias de objetos que saben cómo interactuar con la base de datos, lo que produce un acoplamiento muy fuerte entre ambas capas ya que los objetos del dominio son los responsables de la comunicación con la base de datos.

Por tanto, la decisión final fue aislar la capa de la Lógica del Dominio de la capa de Persistencia añadiendo un nivel de indirección más. Este patrón se conoce como *Data Mapper* y lo que hace es introducir un objeto responsable de la lógica referente a la base de datos (escrituras, consultas, etc) entre la base de datos y la lógica del dominio permitiendo así que ambas capas crezcan independientemente la una de la otra.

Como toda elección, en cuanto a diseño software, el *Data Mapper* tiene sus ventajas, pero también sus desventajas. La desventaja principal es la complejidad que añade introduciendo más objetos, los cuales a medida que crezca nuestro modelo de datos podrían llegar a ser difíciles de mantener. Pero creemos, que la flexibilidad que nos ofrece frente a cambios vale el esfuerzo que hay que invertir.

4.2.2 Aplicando Diseño Orientado a Objetos.

A la hora de instanciar los componentes de los patrones arquitectónicos mencionados se siguió un enfoque orientado a objetos basado en los principios SOLID [22] (*Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency*

inversion). A parte de basarnos en estos principios, en ocasiones hemos utilizado tarjetas CRC (*Candidates, Responsibilities, Collaborators*) como ayuda a la hora de explorar la potencial estructura de objetos de la aplicación. En primer lugar, organizamos el módulo en tres paquetes (service_controllers, domain_objects y data_access) que se correspondían con cada capa del diseño (Figura 4.4).

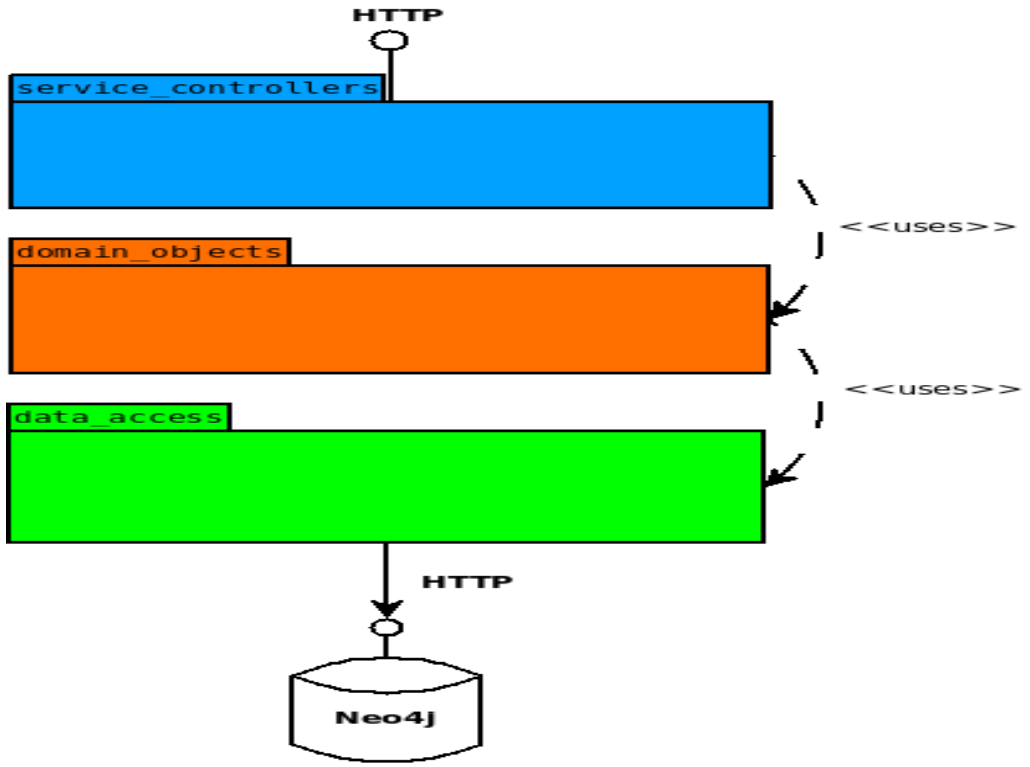


Figura 4.4 *Capas de la arquitectura.*

Diseño de la capa Service Layer.

Esta capa dentro de nuestro *back-end* es la encargada de envolver toda la funcionalidad de nuestra aplicación y ofrecerla a modo de interfaz al exterior.

A la hora de diseñar este nivel solo ha sido necesario establecer el punto de entrada a nuestra aplicación en el módulo *cloudroomservice.coffee*, el cual actúa como *Front Controller* [23]. En Express.js (framework web utilizado) patrón *Front Controller* encaja de forma natural ya que el punto de entrada de una aplicación construida en Express.js alberga toda la lógica de configuración y puesta en marcha de la aplicación. Es decir, la responsabilidad de esta capa pasa al *middleware* encargado del enrutamiento de peticiones HTTP de Express.js, el cual se encarga de exponer la API REST y coordinar todas las peticiones y respuestas, por lo que este punto de entrada tiene como responsabilidad la configuración de todos los *middleware* de Express.js que tienen la lógic común de la aplicación para cada petición.

La recepción de peticiones por parte del punto de entrada será redirigida a la capa *Domain Model*, encargada de la lógica de negocio, de forma asíncrona.

Diseño de la capa Domain Model.

En esta capa se encuentra toda la lógica de negocio, es la encargada de decidir qué hacer con cada entidad concreta del dominio y la conforman las siguientes familias de objetos:

- ***Controllers****:* actúan como *Helpers* del punto de entrada de la aplicación, ayudando con las validaciones de los datos y construcción de respuestas. Su implementación también hace que tengan responsabilidades de *Dispatcher* encargándose del enrutamiento y coordinación de peticiones y respuestas entre capas. A la hora de definir los *Controllers* se hizo encapsulando funcionalidad que compartía un dominio de la aplicación.

- ***Data Transfer Object (DTO)****:* almacenan el estado de las entidades del dominio para compartir dicho estado entre los objetos del sistema. Su responsabilidad es la de sacar las propiedades referentes a las entidades del dominio, tanto de las respuestas por parte de la capa de Persistencia como de las peticiones contra la API REST.

- ***Resource Assemblers***: se encargan de construir la respuesta en un formato JSON+HAL. Para su implementación se decidió aplicar *Dependendy Inversion* usando el patrón *Factory Method* [23] para desacoplar la creación de los recursos y así permitir, por un lado, introducir en un futuro nuevos recursos (como XML) sin que tener muchos efectos colaterales sobre el código, y por otro lado, facilitar los test unitarios permitiendo hacer *mocks* para la creación de objetos. Además, se utilizó en patrón *Composite* [23] para la creación de recursos JSON+HAL que tengan otros recursos embebidos.

Diseño de la capa Data Mapper.

Esta capa ha sido introducida como un nivel de indirección responsable de desacoplar la lógica de negocio de la capa de Persistencia haciéndola transparente a esta. Dentro de este nivel nos encontramos con los siguientes objetos:

- ***Data Access Object (DAO)****:* los *Controllers* acceden a estos objetos mediante un *Factory Method* que les devuelve el DAO necesario dependiendo del dominio de la aplicación que estén tratando, y dicho DAO se encarga de comunicarse con la base de datos que haya por debajo. Al tratarse de un objeto del cual solo es necesaria una instancia, la referencia de este se le pasa a los *Controllers* a modo de *Singleton* [23]. El patrón *Singleton* viene implementado de forma natural en Node.JS, donde el módulo encargado de importar módulos cachea estos, así que solo es necesario importar una instancia del DAO.

4.2.3 Diagrama de clases.

El diagrama (Figura 4.5) muestra las clases que han surgido a partir de las decisiones de diseño tomadas. En un principio el diagrama no mostraba todas las clases que muestra ahora, ya que este ha ido evolucionando tras la fase de *testing* de cada iteración, donde se han realizado refactorizaciones del código. Por tanto, tampoco estamos hablando de que este sea un diseño final, pero el diseño realizado se ha orientado a facilitar cualquier cambio que sea necesario en el futuro.

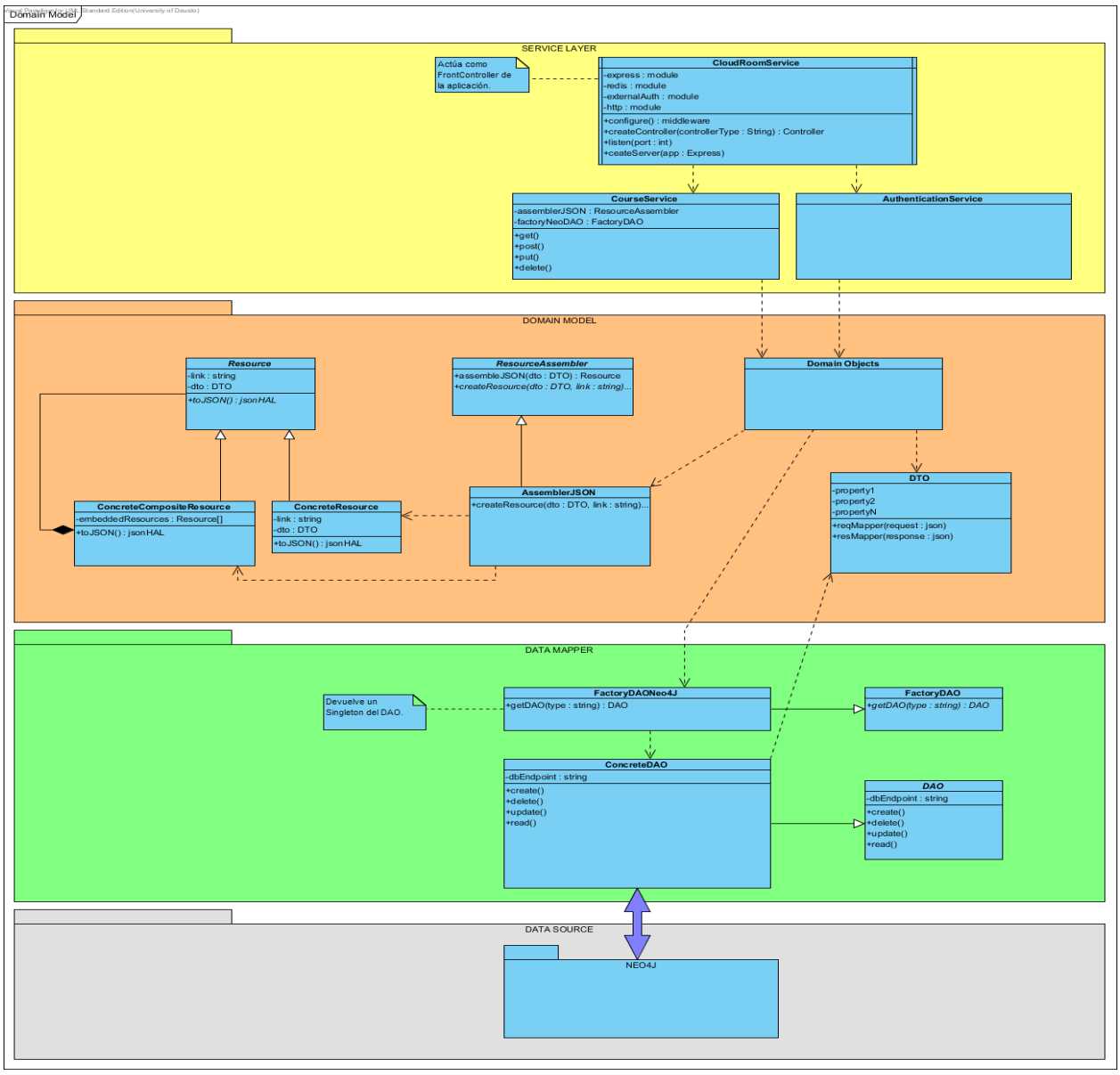


Figura 4.5 *Diagrama de clases del módulo de cursos.*

4.3 Diseño de la API REST.

La elección de una arquitectura RESTful está basada en la necesidad que tiene nuestra aplicación de ser flexible, y aun así mantener en la mayor medida de lo posible un alto rendimiento. Es por esto, que cualidades como la simplicidad y tamaño de mensajes de la arquitectura RESTful se haya impuesto sobre la complejidad y estandarización de la arquitectura SOAP, la cual introduce una mayor QoS (Quality of Service), pero que debido a su complejidad introduce mucha más sobrecarga. Teniendo esto en mente, y que la aplicación tendrá una mayor carga en cuanto a lecturas que escrituras, hemos creído que los beneficios en cuanto a seguridad y disponibilidad que ofrece SOAP no eran los que buscábamos en nuestra aplicación, donde están por delante otros como rendimiento.

Como ya hemos mencionado una de las grandes debilidades de la arquitectura RESTful reside en la poca estandarización, lo que ha llevado a que muchas APIs una vez publicadas no puedan ser modificadas. Por este motivo, diseñaremos la API de modo que sea auto-contenida, que sea navegable, cumpliendo así el principio HATEOAS (*Hypermedia as the Engine of Application State*) de la arquitectura RESTful [24], y que sea lo más estandarizada posible.

4.3.1 Elección de Media –Type.

A la hora de escoger con que media-type dialogaría nuestra API, nos decantamos por JSON debido a su naturaleza compacta y fácilmente legible, además de ser más veloz en la transferencia de datos que XML, ya que hay que transferir menos datos y a que el *parsing* de estos es más veloz puesto que el *back-end* de la aplicación está implementado en JavaScript.

Por contra, JSON no es un formato *hypermedia*, así que hemos considerado otros formatos como Collection+JSON y HAL (Hypertext Application Language).

Finalmente, nos hemos decantado por HAL [25] por su sencillez a la hora de hacer la API explorable, permitiendo *links* altamente personalizables y una documentación fácil de encontrar desde el propio mensaje. Por tanto, el poder de este formato se asienta sobre su sencillez y la documentación que se ligue a la API.

HAL – Hypertext Application Lenguage

Este formato ofrece convenciones para expresar *hyperlinks* tanto en JSON como XML. Esto nos da la posibilidad de incluir XML en el futuro si fuese necesario. Es un formato sencillo con las siguientes características.

- **Content – Type:** application/hal+json
- **Modelo:**
    - *Resources:* tienen *links* (a URIs), *embedded resources* (otros recursos contenidos dentro del mismo) y *state* (datos del JSON).
    - *Links:* tiene un *target* (URI), una relación 'rel' (nombre del *link*).

A continuación se muestra una imagen con la abstracción del modelo, donde se puede apreciar la recursividad del formato (Figura 4.6).

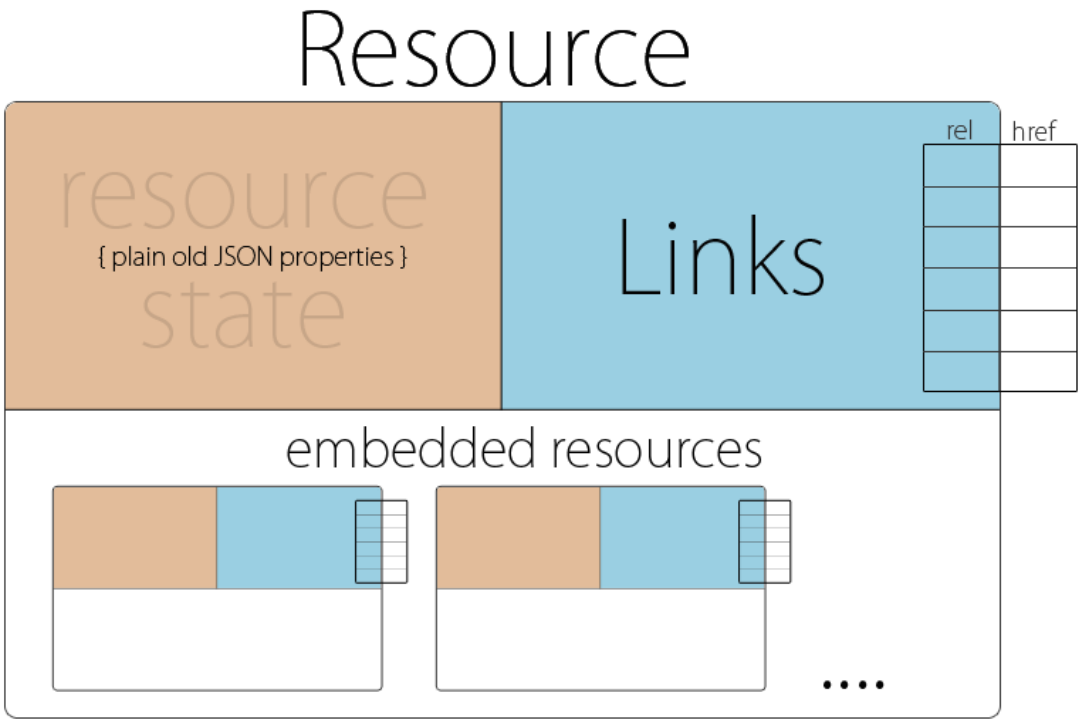


Figura 4.6 *Abstracción del formato JSON+HAL.*

4.3.2 Descripción de la API.

En esta sección se contemplan una serie de tablas descriptivas a modo de documentación, para dotar de la semántica de nuestra aplicación a la API.

Operación	Listados de cursos
Método	GET
Recurso	
URI	/courses (?start=[start]) (?end=[end])
Cadena de consulta	
Devuelve	200 OK + (application/hal+json)
	404 NOT FOUND 400 BAD REQUEST

Operación	Información básica de un curso
Método	GET
Recurso	
URI	/courses/{id}
Cadena de consulta	
Devuelve	200 OK + (application/hal+json)
	404 NOT FOUND 400 BAD REQUEST

Operación	Obtener los anuncios de un curso.
Método	GET
Recurso	
URI	/courses/{id}/{session}/news
Cadena de consulta	
Devuelve	200 OK + (application/hal+json)
	404 NOT FOUND 400 BAD REQUEST

Operación	Obtener los anuncios filtrados por categoría y/o cantidad.
Método	GET
Recurso	
URI	/courses/{id}/{session}/news?type={type}&start={start}
Cadena de consulta	
Devuelve	200 OK + (application/hal+json)
	404 NOT FOUND 400 BAD REQUEST

Operación	Obtener lecciones y sus capítulos ordenados.
Método	GET
Recurso	
URI	/courses/{id}/{session}/lessons
Cadena de consulta	
Devuelve	200 OK + (application/hal+json)
	404 NOT FOUND 400 BAD REQUEST

Operación	Obtener contenido de un capítulo.
Método	GET
Recurso	
URI	/courses/{id}/{session}/lessons/{id}/chapters/{id}
Cadena de consulta	
Devuelve	200 OK + (application/hal+json)
	404 NOT FOUND 400 BAD REQUEST

Operación	Obtener vídeos de un curso.
Método	GET
Recurso	
URI	/courses/{id}/{session}/videos
Cadena de consulta	
Devuelve	200 OK + (application/hal+json)
	404 NOT FOUND 400 BAD REQUEST

Operación	Obtener un quiz de un vídeo.
Método	GET
Recurso	
URI	/courses/{id}/{session}/videos/{id}/quizs/{id}
Cadena de consulta	
Devuelve	200 OK + (application/hal+json)
	404 NOT FOUND 400 BAD REQUEST

Operación	Comprobar respuestas de un quiz.
Método	POST
Recurso	
URI	/courses/{id}/{session}/videos/{id}/quizs/{id}
Cadena de consulta	{answer}
Devuelve	200 OK + (application/hal+json)
	404 NOT FOUND 400 BAD REQUEST

Operación	Obtener los tests de un curso.
Método	GET
Recurso	
URI	/courses/{id}/{session}/tests
Cadena de consulta	
Devuelve	200 OK + (application/hal+json)
	404 NOT FOUND 400 BAD REQUEST

Operación	Obtener un test de un curso.
Método	GET
Recurso	
URI	/courses/{id}/{session}/tests/{id}
Cadena de consulta	
Devuelve	200 OK + (application/hal+json)
	404 NOT FOUND 400 BAD REQUEST

Operación	Comprobar respuestas de un test.
Método	POST
Recurso	
URI	/courses/{id}/{session}/tests/{id}
Cadena de consulta	{[questionID, answer]}
Devuelve	200 OK + (application/hal+json)
	404 NOT FOUND 400 BAD REQUEST

Operación	Obtener el progreso en un curso.
Método	GET
Recurso	
URI	/courses/{id}/{session}/progress
Cadena de consulta	
Devuelve	200 OK + (application/hal+json)
	404 NOT FOUND 400 BAD REQUEST

4.4 Diseño del Modelo de Datos.

La flexibilidad de una base de datos de grafos, la cual prácticamente carece de *schema*, nos ha permitido que el modelado de los datos haya ido evolucionando en paralelo al desarrollo de la aplicación. Es decir, el modelado de datos no ha sufrido apenas impactos por haber seguido una metodología iterativa para desarrollar la aplicación, y cada vez que ha sido necesario refactorizar el diseño, nos ha llevado poco tiempo.

A la hora de diseñar el modelo de datos en Neo4j, lo que dibujas en papel o en pizarra acaba siendo el modelo en sí. Terminas tomando dicho diseño y transfiriéndolo según fue dibujado a la base de datos, lo cual es apoyado por la expresividad del lenguaje de consultas *Cypher*.

4.4.1 Modelado iterativo.

Hemos tratado de encajar el diseño del modelo de datos en un desarrollo de software basado en metodologías ágiles [26]. Por tanto, el modelo de datos ha ido surgiendo poco a poco con pequeñas refactorizaciones a medida que se incluían nuevas funcionalidades a la aplicación. Es decir, el modelo se ha descrito en términos de lo que la aplicación necesitaba, los cuales llegan descritos en forma de historias de usuario dándonos una perspectiva de lo que el usuario necesita.

Partiendo de las historias de usuario, somos capaces de obtener una perspectiva de las consultas que pudieran ser necesarias, y es de esta forma como han surgido los diseños. Hemos trasladado las 'cosas' del mundo real como nodos y les hemos dado un contexto semántico estructurando los nodos mediante relaciones.

Todo el modelado se ha realizado teniendo siempre en mente tanto los requisitos funcionales (historias de usuario), como los principales requisitos no-funcionales (rendimiento y flexibilidad) recogidos del análisis de requisitos.

El diseño final, por tanto, ha quedado recogido en varias partes en las que se tomaron algunas decisiones de diseño importantes y las cuales recogemos a continuación.

Contenido del curso.

El contenido del curso se estructuro de forma jerárquica en forma de árbol, donde cada curso contenía lecciones, capítulos, y vídeos y lecturas en cada capítulo además de un tests para evaluar cada lección (Figura 4.7).

Estructurar cada contenido como un nuevo nodo, en lugar de incorporarlo como propiedades de los nodos lección, surge a raíz de que en Neo4j no existen los tipos complejos y estos deben ser modelados en varios nodos.

Por otro lado, era necesario conocer el orden de las lecciones, y lo que hicimos fue dotar de una propiedad 'índice' a cada lección y capítulo dentro de la lección.

Modelar esta parte del modo que se ha dicho arriba potencio la flexibilidad en detrimento del rendimiento en las consultas, pues una consulta tendría que realizar un recorrido en el que visitaría todos los nodos y compararía sus índices, pero como la historia de usuario necesitaba devolver todo el contenido, todos los nodos se iban a visitar de todas formas. Esta es la razón por la cual decidimos hacer un diseño jerárquico favoreciendo así, la flexibilidad frente a cambios o introducción de contenido.

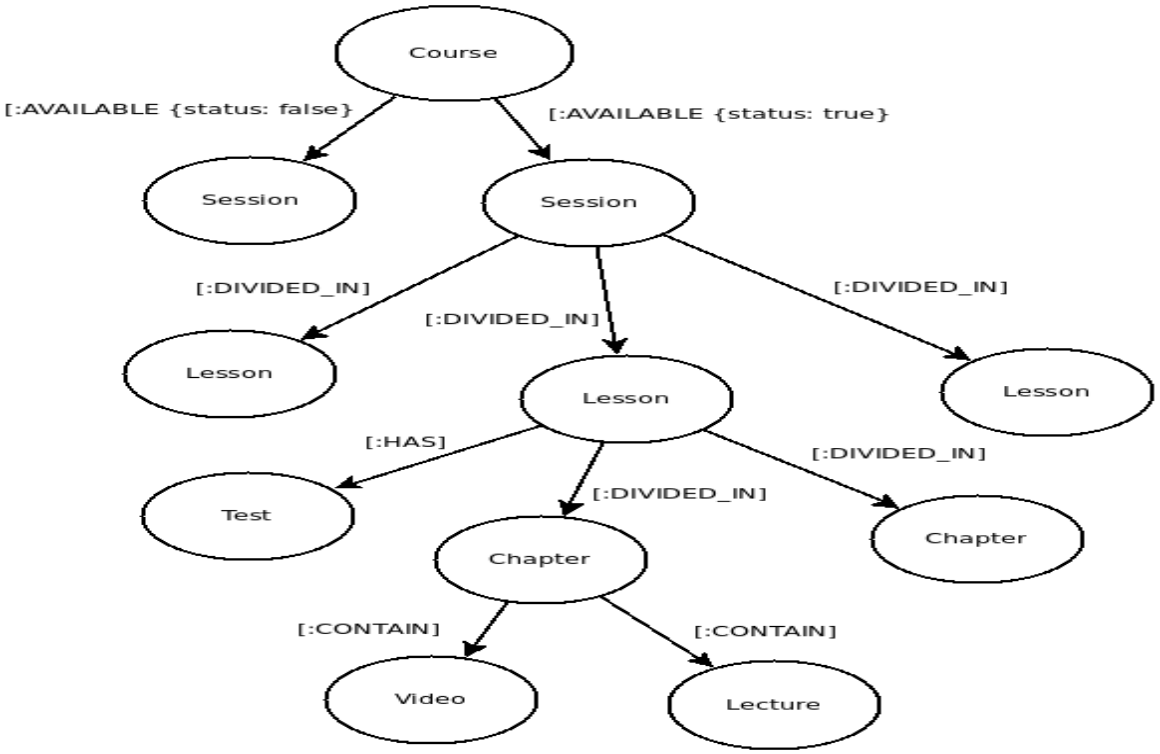


Figura 4.7 *Modelado jerárquico de los contenidos del curso.*

Anuncios del curso.

Los anuncios debían poder ser filtrados por tipos y devueltos empezando por los más recientes (Figura 4.8).

Por otra parte, hemos pensado que los anuncios podrían alcanzar un gran número, por lo que finalmente decidimos diseñar esta parte del modelo de datos como una lista enlazada, que a su vez contiene listas enlazadas dependiendo del tipo de anuncio. Esto mejora el rendimiento en las consultas sobre los anuncios, ya que el recorrido no tiene que acceder a todos los nodos anuncios. Solo tiene que acceder a los N primeros anuncios que se desean devolver.

Tener relaciones hacia los anuncios previos ([:PREVIOUS_SAME_TYPE]) del mismo tipo, construyendo así, listas enlazadas dentro de la lista de anuncios general, se hace debido a que consultar una propiedad dentro de una relación en un recorrido implica un acceso de entrada/salida, ya que las propiedades se almacenan en ficheros separados. Por tanto, esto supone perder mucha velocidad en la consulta.

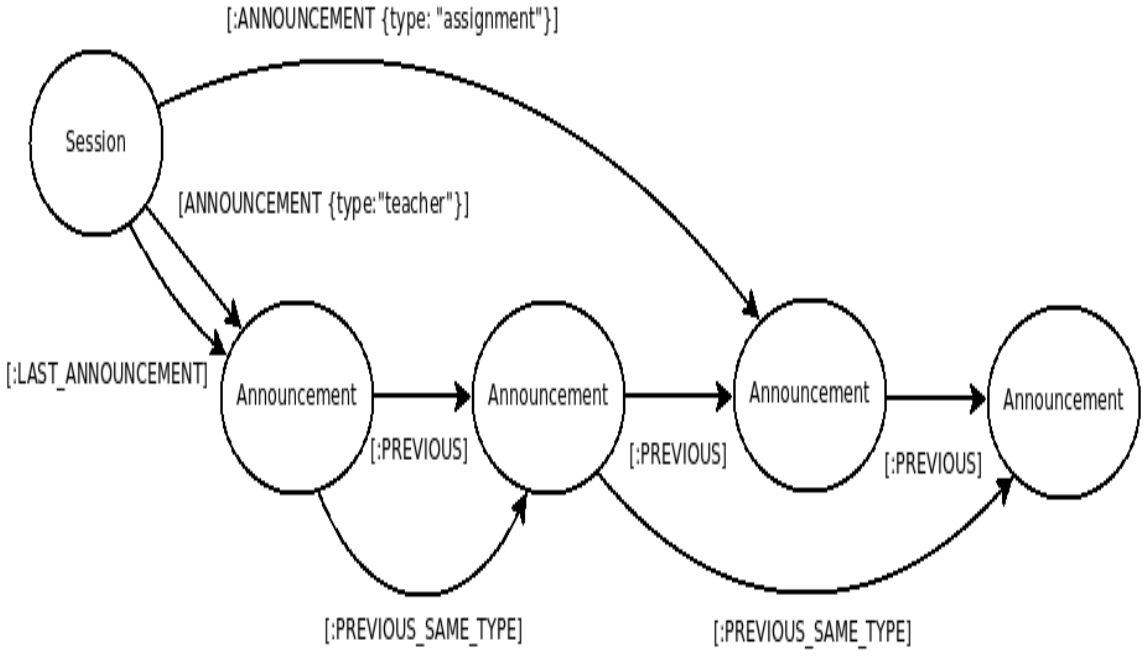


Figura 4.8 *Modelado de las noticias del curso como lista enlazada.*

Progreso del alumno en el curso.

Se trata de consultas para conocer los contenidos superados por parte del alumno y fechas en las que lo hizo (Figura 4.9).

Decidimos modelar la interacción del alumno con el curso como una entidad propia que represente dicha interacción. Este nuevo nodo que surge de la interacción entre alumno y curso representa el resultado de dicha interacción.

Esto dota de flexibilidad en la estructura del grafo y nos permite evolucionar el progreso para añadir nuevos elementos a monitorizar en la interacción del alumno con el curso.

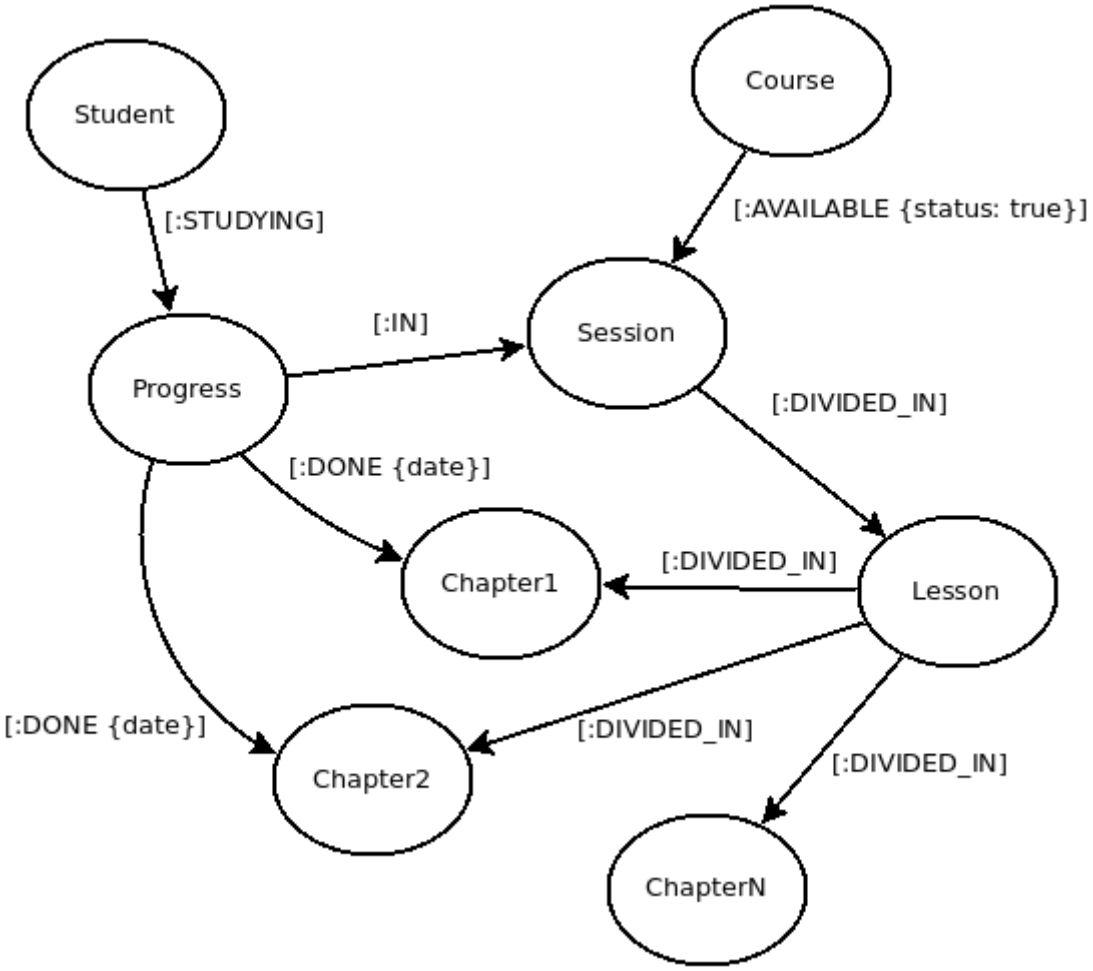


Figura 4.9 *Modelado del progreso en el curso como nodo que recoge la interacción alumno-curso.*

Realización de tests y cuestiones embebidas en vídeos.

En el curso se ofrece la posibilidad de realizar test evaluativos de cada lección, los cuales pueden contener varias cuestiones, y cuestiones embebidas en vídeos que funcionan para registrar el seguimiento del curso. Cada una de las cuestiones ya sean embebidas o no, se registran en el modelo de datos de igual manera, con la excepción de que los tests tienen una propiedad que indica el peso de las cuestiones, y las cuestiones embebidas una propiedad que permite saltar en el vídeo a la explicación de la cuestión (Figura 4.10).

Por no existir los tipos compuestos, hemos tenido que modelar las respuestas a las cuestiones aparte.

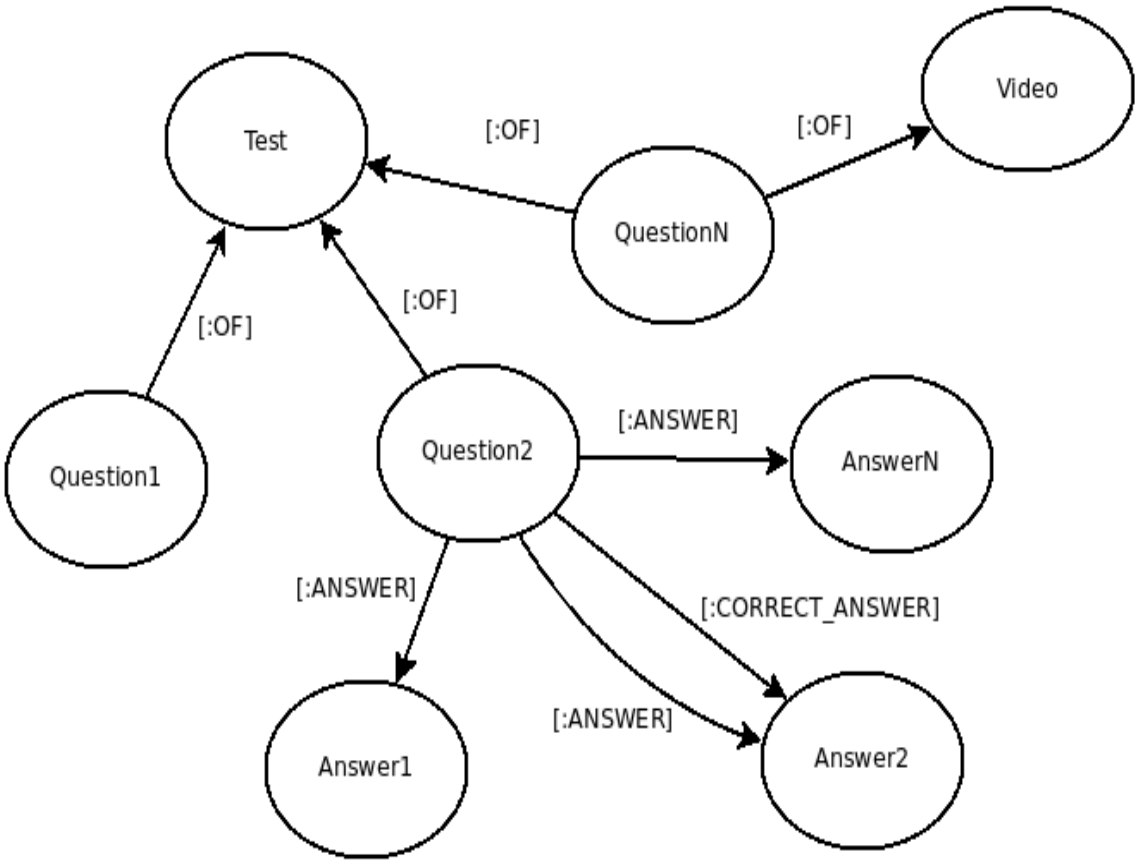


Figura 4.10 *Modelado de las cuestiones en vídeos y tests.*

4.4.2 Modelo de Datos.

Finalmente nuestro modelo de datos presenta la siguiente estructura y propiedades. Este modelo de datos no es definitivo y está preparado para refactorizaciones debido a cambios en los requisitos (Figura 4.11).

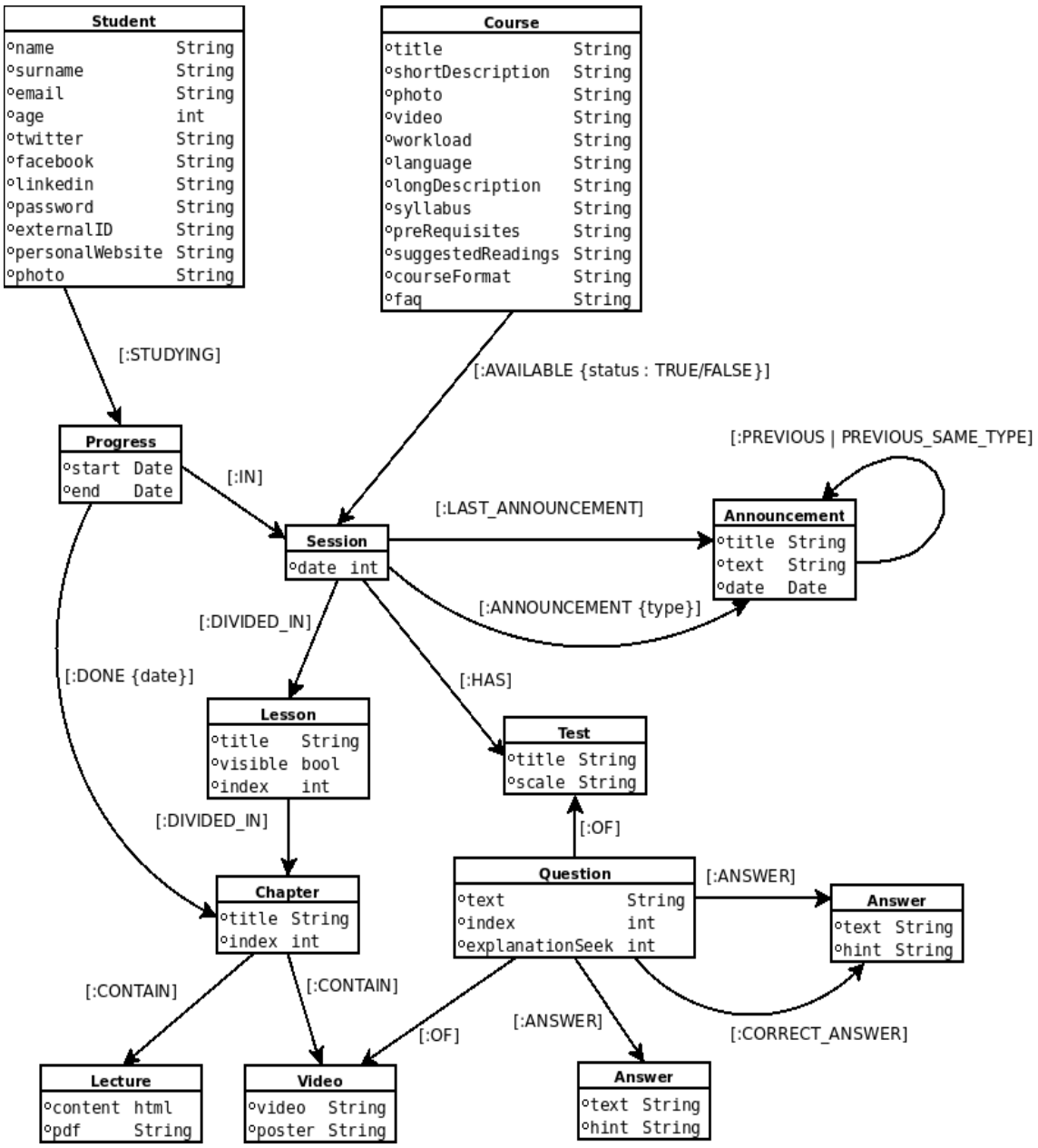


Figura 4.11 *Modelo de datos del Módulo de Cursos.*

REFLEXIÓN

“La información no es conocimiento.”

Albert Einstein

5 Conclusiones.

Este proyecto ha conseguido definir un módulo englobado dentro de una arquitectura mayor con todas las restricciones que esto supone.

Partiendo de un diseño arquitectónico de alto nivel y con el objetivo de instanciar cada uno de sus componentes software, se estableció un grupo de desarrolladores y una metodología de desarrollo ágil para su consecución. Esto supuso, una primera toma de contacto con un ámbito de desarrollo auto-organizado y cooperativo, done la experiencia de las personas jugó un papel fundamental.

Por tanto, he sido capaz de identificar, no solo aquellos aspectos del proyecto referentes al diseño e implementación de los componentes software, sino también los referentes a la gestión y organización de equipos.

Por esta razón, los resultados de este proyecto se pueden considerar en diferentes campos: organización, implementación y revisión. Madurar cada uno de estos aspectos ha sido fundamental para el éxito del proyecto.

La organización, evolucionó, sobre todo, durante el primer mes. Este mes, supuso una puesta en marcha de las metodologías y herramientas de gestión que íbamos a utilizar. Pudimos comprobar como la integración de nuevos tipos de forma de trabajo supone un gran retraso en el desarrollo del producto, pero más tarde supimos darnos cuenta de las ventajas que esta integración supuso, automatizando cada vez este tipo de tareas.

La implementación bajo una serie de restricciones previas, nos hizo darnos cuenta de que no existe la decisión perfecta en cuanto a diseño, sino que hay una decisión que encaja en cada situación. Además, al intentar integrar tu módulo en una arquitectura mayor logras ver cómo el número de dependencias de tu módulo con el resto del sistema reduce la productividad.

Finalmente, las revisiones periódicas de lo hecho, permite obtener un *feedback* muy útil para no repetir los mismos errores. De hecho, pudimos apreciar, como este tipo de revisiones y la capacidad analítica en este punto puede suponer la mayor diferencia entre el éxito o fracaso del proyecto.

A lo largo de estos 4 meses he podido comprobar, cómo toda la información recogida a lo largo de estos 4 años de carrera se ha visto dotada por una gran carga semántica. Esto ha sido gracias a la oportunidad que me ha brindado este proyecto, donde he tenido que

aplicar todos mis conocimientos a la hora de soslayar los obstáculos que han ido apareciendo durante su desarrollo.

Con este proyecto, por tanto, he salido de mi zona de confort para adentrarme en los eriales de mi propia ignorancia. Esto ha logrado que la información que poseía se convirtiese en conocimiento, y que haya descubierto nuevos caminos por explorar en el campo de la informática.

6 Líneas futuras.

Durante el desarrollo del módulo de cursos hemos identificado una serie de posibilidades que decidimos no introducir en el momento y dejarlas para más adelante, en esta sección hablaremos de cada una de ellas.

6.1 Documentación auto-descubrible en la API REST.

La idea es complementar el formato elegido (JSON+HAL) con una documentación que dote de una fuerte carga semántica a cada recurso y publicar dicha documentación para que los desarrolladores que quieran consumir nuestra API tengan mucha facilidad para ello. Por tanto, nuestra idea sería que cada recurso en su cabecera tuviese, además, una referencia a la API a modo de *link*, permitiendo así una navegación total a través de la API, haciendo que incluso la documentación sea auto-descubrible.

6.2 TDD (*Test-Driven Development*).

Buscando aumentar la productividad en el desarrollo, creemos que integrar poco a poco, debido a su dificultad y la poca experiencia en TDD de los miembros del equipo, esta metodología, supondrá en un futuro una gran ventaja que eliminará muchos de los errores de la aplicación y nos permitirá automatizar todavía más el proceso que va desde el diseño hasta la puesta en producción.

Para ello, TDD se basa en escribir tests en primer lugar, implementar contra los test, y refactorizar a continuación (Figura 6.1). De este modo van surgiendo poco a poco las buenas decisiones de diseño.

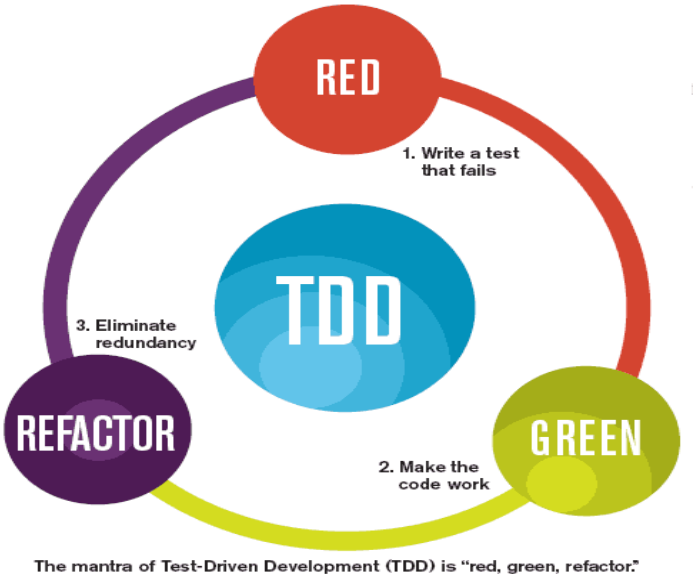


Figura 6.1 *Pasos del TDD.*

6.3 Funcionalidades gamificadas.

Pensamos que las funcionalidades gamificadas estarían muy valoradas por parte de los alumnos y ayudaría a evitar la tasa de abandonos en los cursos.

Este tipo de funcionalidades están siendo adoptadas por muchas instituciones de enseñanza, y muchos estudios aseguran que este tipo de funcionalidades ayudan al aprendizaje.

Apéndice A:
Pensamientos del autor.

¡AVISO AL LECTOR!: Estas líneas son crudas y sin edición, queda bajo la responsabilidad del lector la decisión de aventurarse en ellas.

La última vez que recuerdo haberme enfrentado a un desafío como el que ha supuesto este proyecto, fue hace mucho, mucho tiempo, en una galaxia muy, muy lejana…

Por aquel entonces todavía tenía que andar buscando la marca verde que señalaba la película como apta para todos los públicos en los videoclubs, sí, has oído bien, videoclubs… aquellos sitios plagados de VIDEOCASETES!!!. Pues bien, en aquel entonces me encontré con una serie de monstruos de bolsillo que llenaron mis tardes durante un buen tiempo. Y fue con ellos con los que tuve que superar uno de los mayores desafíos que recuerdo. Se trataba de superar la Liga Pokemon!!!!

Ya había invertido sudor y sangre en superar todos y cada uno de los gimnasios pokemon previos a la liga, y la liga final no fue una excepción a la hora de poner a prueba mis habilidades como entrenador pokemon, pero finalmente y después de un gran esfuerzo logre superarla… oh, todavía recuerdo aquella música celestial de 8-bits tras la victoria.

Aquel reto fue duro, pero tras él me di cuenta que aún me quedaban más retos por superar… como hacerse con los 150 pokemon !!!!!

Y hoy en día, como en aquel entonces, y gracias a este proyecto, me he dado cuenta de todo el esfuerzo que he invertido y todo lo que he aprendido. Y otra vez, como en aquel entonces, lejos de sentirme más sabio, me he sentido más ignorante, porque he visto el camino y todo lo que me queda por recorrer.

Bibliografía

[1] Wikipedia. (2014) wikipedia.org. [Online]. http://en.wikipedia.org/wiki/Cloud_computing

[2] Gartner. (2008, Agosto) www.gartner.com. [Online]. http://www.gartner.com/newsroom/id/739613

[3] Wikipedia. (2014) wikipedia.org. [Online]. http://en.wikipedia.org/wiki/Douglas_Parkhill

[4] Wikipedia. (2014) wikipedia.org. [Online]. http://en.wikipedia.org/wiki/Massive_open_online_course

[5] McKinsey&Company, "Education to Employment: Designing a System that Works," 2014.

[6] Coursera. (2014) www.coursera.org. [Online]. www.coursera.org

[7] edX. (2014) www.edX.org. [Online]. https://www.edx.org/

[8] Open edX. (2014) code.edx.org. [Online]. http://code.edx.org/

[9] Alberto Rojo. (2013, Julio) www.pistel.com. [Online]. http://www.psitel.com/elearning/?p=920

[10] Udacity. (2014) www.udacity.com. [Online]. https://www.udacity.com/courses#!/All

[11] Udacity. (2013, Enero) blog.udacity.com. [Online]. http://blog.udacity.com/2013/01/site-redesign-updated-design-and.html

[12] Moodle. (2014) docs.moodle.org. [Online]. http://docs.moodle.org/26/en/Features

[13] Tom Hughes-Croucher & Mike Wilson, *Node: Up and Running*.: O'Reilly, 2012.

[14] Pedro Teixeira, *Professional Node.js*.: John Wiley & Sons, Inc., 2013.

[15] Neo Database, "The Neo Database - A Technology Introduction," 2006.

[16] Neo Technology, "The Neo4j Manual," 2013.

[17] CloudRoom. CloudRoom. [Online]. http://cloudroom.co/

[18] Francisco Javier Sánchez Carmona, *TFG. CloudRoom: Una Plataforma Social, Semántica*

y Ubicua., 2014.

[19] Paul Clements, Rick Kazman Len Bass, *Software Architecture in Practice*, 3rd ed.: Addison–Wesley, 2012.

[20] Henrik Kniberg, *Scrum y XP desde las trincheras*.: InfoQ, 2007.

[21] Martin Fowler, *Patterns of Enterprise Application Architecture*.: Addison-Wesley, 2002.

[22] Nat Pryce Steve Freeman, *Growing Object-Oriented Software, Guided by Tests*.: Addison-Wesley, 2009.

[23] Richard Helm, Ralph Johnson, John M.Vlissides Erich Gamma, *Design Patterns: Elements of Reusable Object-Oriented Software*.: Addison-Wesley, 1994.

[24] Mike Amundsen Leonard Richardson, *RESTful Web APIs*.: O'REILLY, 2013.

[25] Mike Kelly. stateless.co. [Online]. http://stateless.co/hal_specification.html

[26] Jim Webber, Emil Eifrem Ian Robinson, *Graph Databases*.: O'REILLY, 2013.

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
	Fecha/Hora	Fri Jun 06 23:19:56 CEST 2014
	Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
	Numero de Serie	630
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)